INTERSECTING LINE-SEGMENTS

Vera Sacristán

Discrete and Algorithmic Geometry
Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

INTERSECTING LINE-SEGMENTS

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Some applications (in addition to those that you already know)

Geographic information systems
Detecting the intersections among the elements of the different layers of information (cities, roads, services, ...)

Realistic visualization
Eliminating the hidden portions of a scene

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line-segments. This algorithm runs in $\Theta(n^2)$ time.

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line-segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line-segment configurations with $\binom{n}{2}$ intersections.

Problem

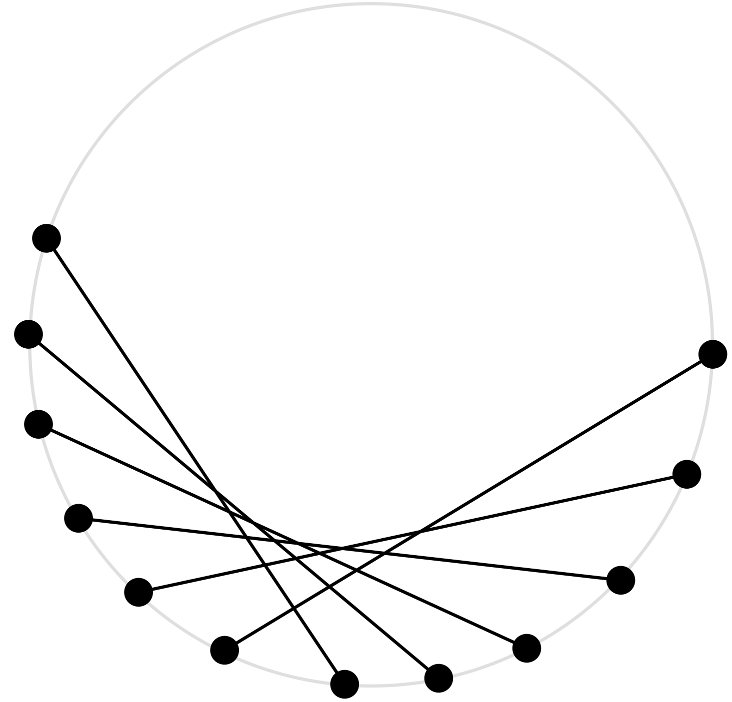
Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line-segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line-segment configurations with $\binom{n}{2}$ intersections.



INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

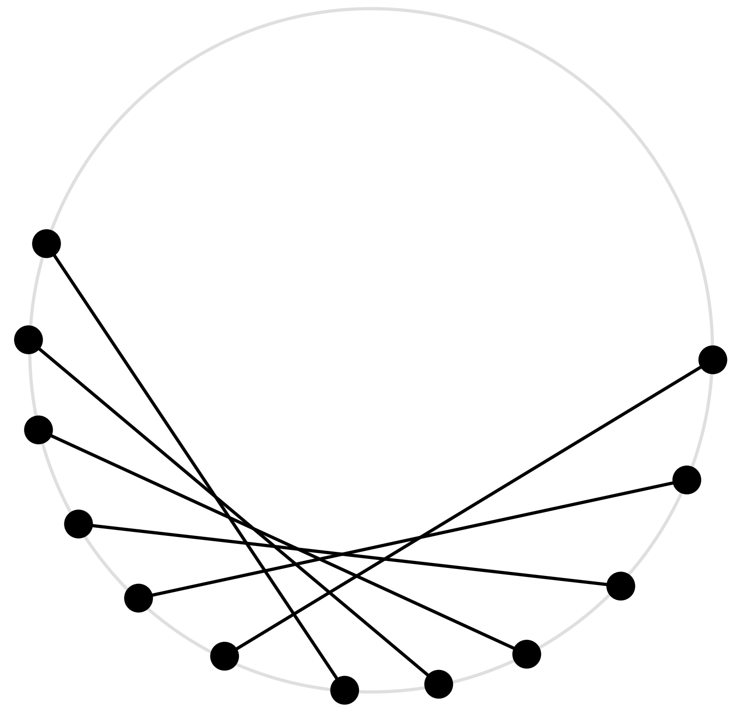
Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line-segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line-segment configurations with $\binom{n}{2}$ intersections.

Nevertheless, there exist sets of n line-segments with a number of intersections substantially smaller than $\binom{n}{2}$.



INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

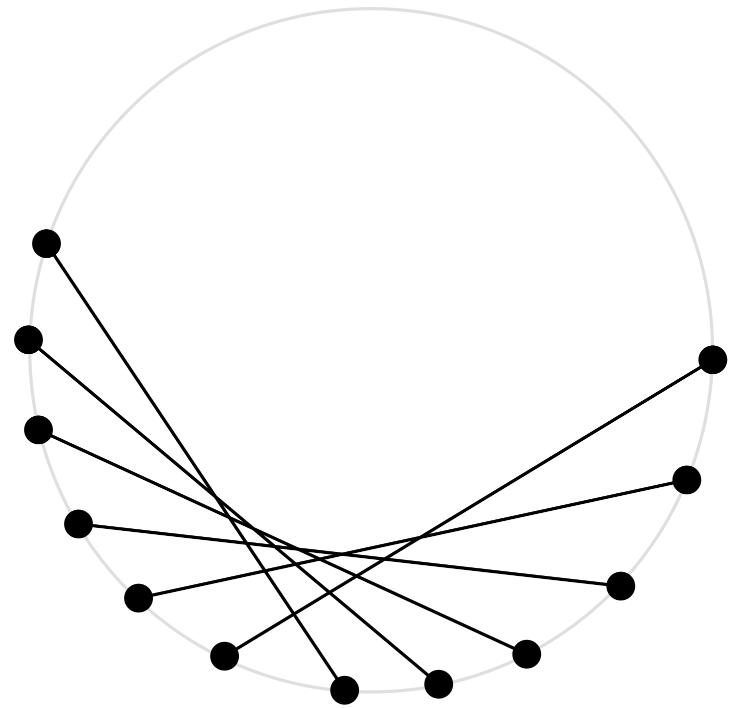
Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line-segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line-segment configurations with $\binom{n}{2}$ intersections.

Nevertheless, there exist sets of n line-segments with a number of intersections substantially smaller than $\binom{n}{2}$.



Output-sensitive solution

Algorithm whose running time depends on the number of intersections to be reported.

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Solution

Sweep line algorithm

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Solution

Sweep line algorithm

A straight line (vertical, in this case) scans the object (the line segments) and allows detecting and computing the desired elements (intersection points), leaving the problem solved behind it. The sweeping process is discretized.

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Solution

Sweep line algorithm

A straight line (vertical, in this case) scans the object (the line segments) and allows detecting and computing the desired elements (intersection points), leaving the problem solved behind it. The sweeping process is discretized.

Essential elements of a sweep line algorithm:

- Sweep line

  Data structure storing the information of the objects intersected by the sweeping line, sorted along the line.

- Events queue

  Priority queue keeping the information of the algorithm stops, i.e., all the positions where the sweep line structure changes.

INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

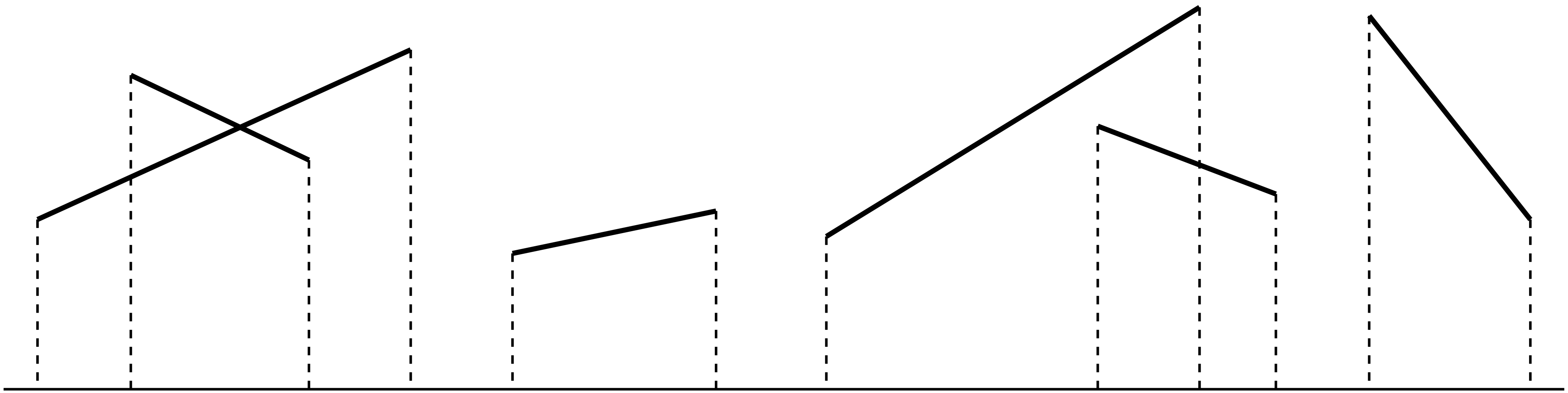
INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.



INTERSECTING LINE-SEGMENTS

Problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line-segments with a line, two intersecting line-segments need to be consecutive in the sweep line right before their intersection point.

INTERSECTING LINE-SEGMENTS

Problem

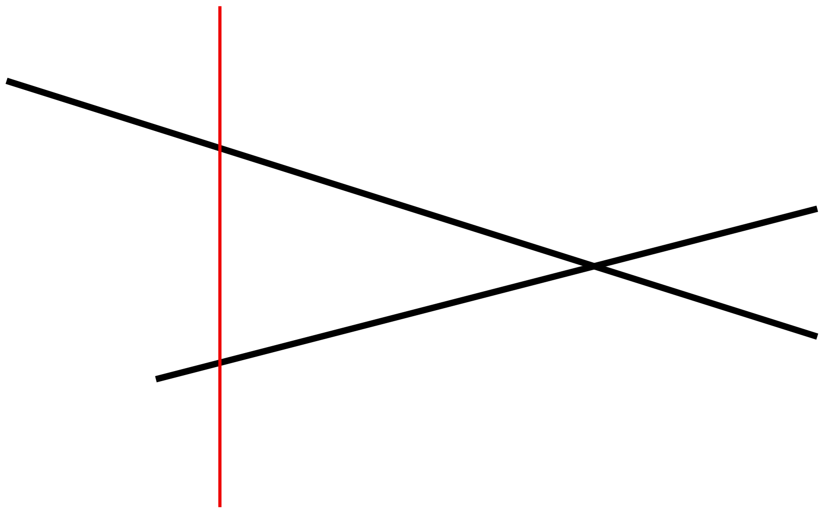
Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line-segments with a line, two intersecting line-segments need to be consecutive in the sweep line right before their intersection point.



INTERSECTING LINE-SEGMENTS

Problem

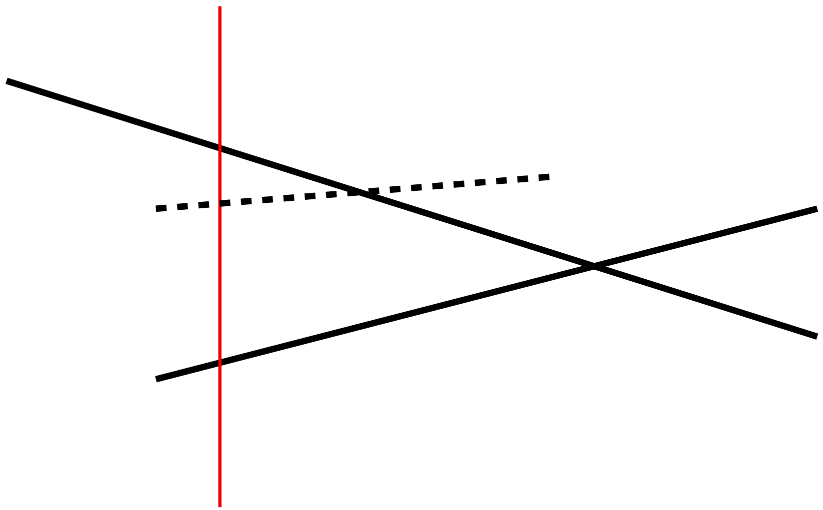
Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line-segments with a line, two intersecting line-segments need to be consecutive in the sweep line right before their intersection point.



INTERSECTING LINE-SEGMENTS

Problem

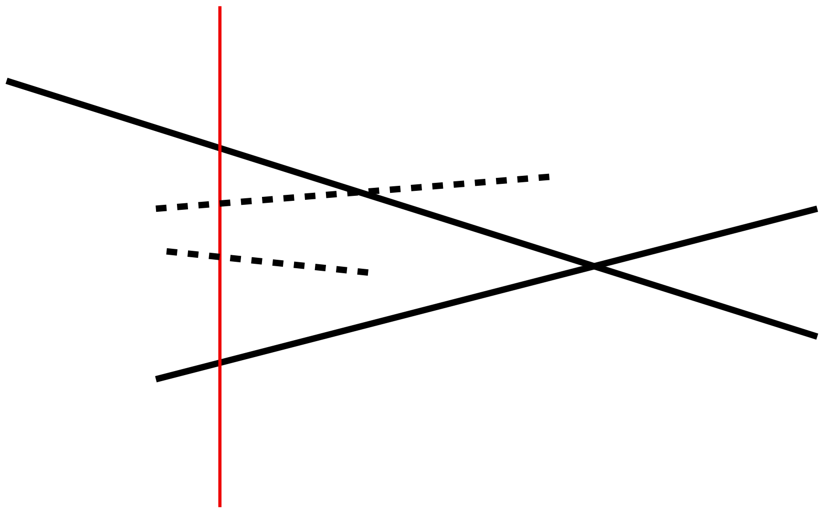
Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line-segments with a line, two intersecting line-segments need to be consecutive in the sweep line right before their intersection point.



INTERSECTING LINE-SEGMENTS

Problem

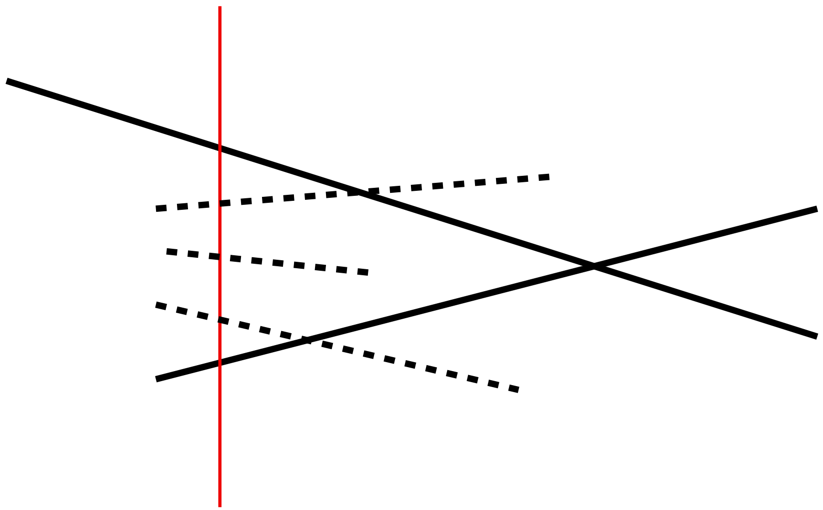
Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Observation 1

If two line-segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line-segments with a line, two intersecting line-segments need to be consecutive in the sweep line right before their intersection point.



Bentley-Ottman's algorithm

This is a sweep-line algorithm.

Bentley-Ottman's algorithm

This is a sweep-line algorithm.

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line-segments, and no two endpoints of two line-segments, no two intersection points of line-segments, no endpoint and intersection point, lie in the same vertical line.
2. At each intersection point, only two line-segments intersect.

Bentley-Ottman's algorithm

This is a sweep-line algorithm.

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line-segments, and no two endpoints of two line-segments, no two intersection points of line-segments, no endpoint and intersection point, lie in the same vertical line.

2. At each intersection point, only two line-segments intersect.

Sweep line

Stabbed line-segments, in vertical order.

Bentley-Ottman's algorithm

This is a sweep-line algorithm.

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line-segments, and no two endpoints of two line-segments, no two intersection points of line-segments, no endpoint and intersection point, lie in the same vertical line.

2. At each intersection point, only two line-segments intersect.

Sweep line

Stabbed line-segments, in vertical order.

Events

- All endpoints of the line-segments (known a priori).
- All intersection points of line-segments (found on the fly).

Bentley-Ottman's algorithm

Bentley-Ottman's algorithm

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Bentley-Ottman's algorithm

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: the $k = O(n^2)$ intersections of line-segment pairs, (x, y, i, j).

Initialization:
- Sort the $2n$ endpoints by abscissa and store the information in E.
- Line L starts empty.

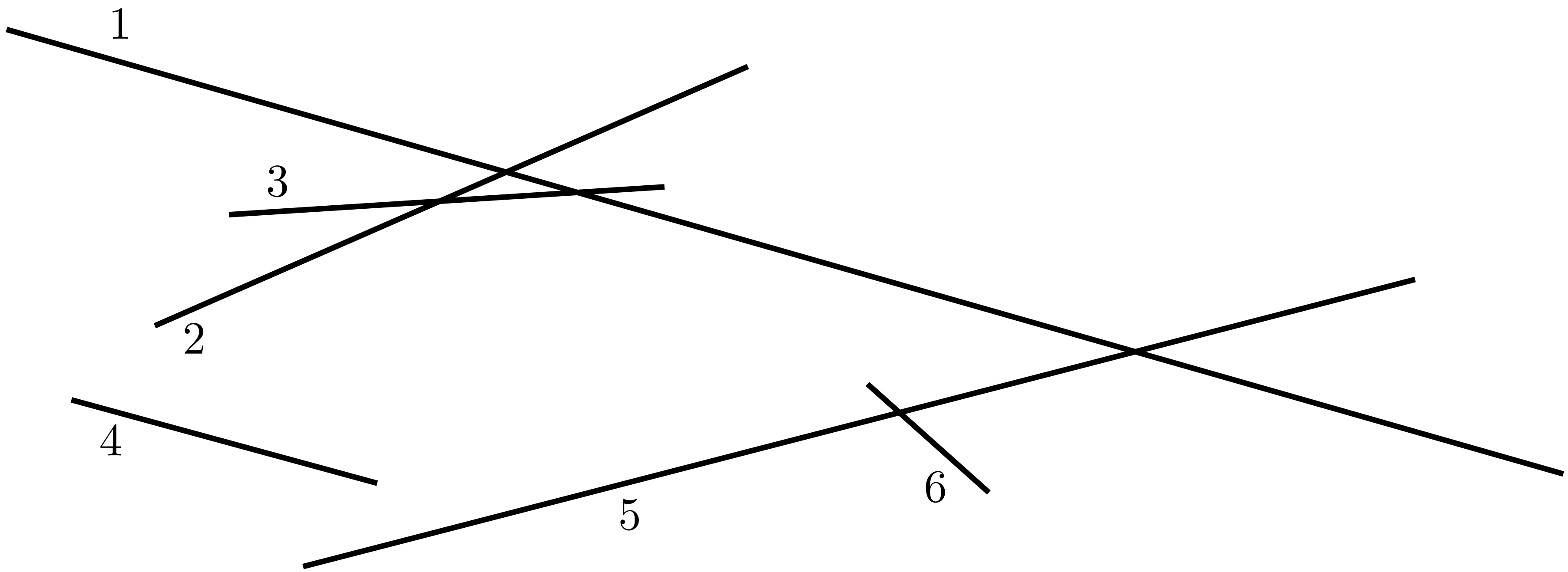
INTERSECTION OF LINE-SEGMENTS

Advance

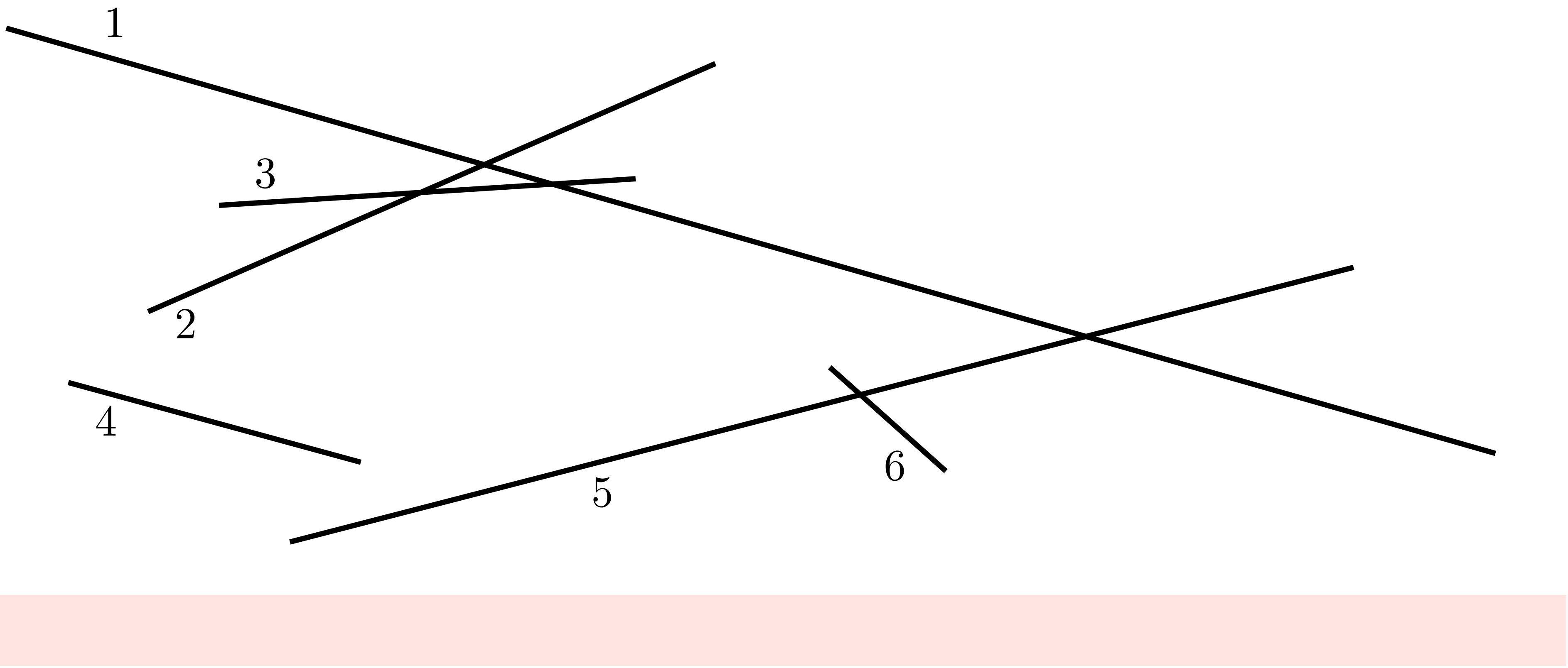
While $E \neq \emptyset$ do:

1. $p = \min E$
2. If $p = \text{start}(s)$, then:
   - Insert s in L
   - If s^- and s intersect to the right of p, insert their intersection point in E and report it (if needed). Do the same for s^+.
3. If $p = \text{end}(s)$, then:
   - If s^- and s^+ intersect to the right of p, insert their intersection point in E and report it (if needed).
   - Delete s from L
4. If $p = s_1 \cap s_2$ with $s_1 <_L s_2$, then:
   - If s_1^- and s_2 intersect to the right of p, insert their intersection point in E and report it (if needed). Do the same for s_2^+ and s_1.
   - Transpose s_1 and s_2 in L
5. Delete p from E

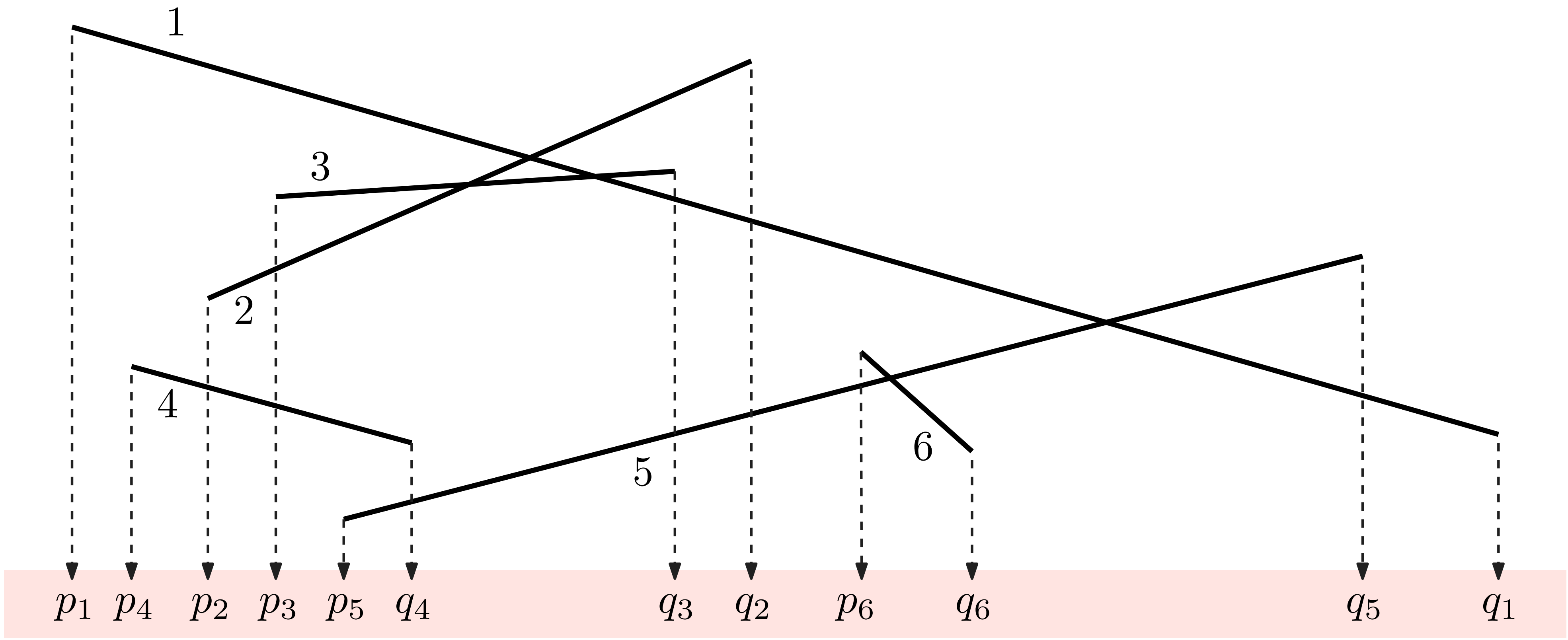
Bentley-Ottman's algorithm: simulation



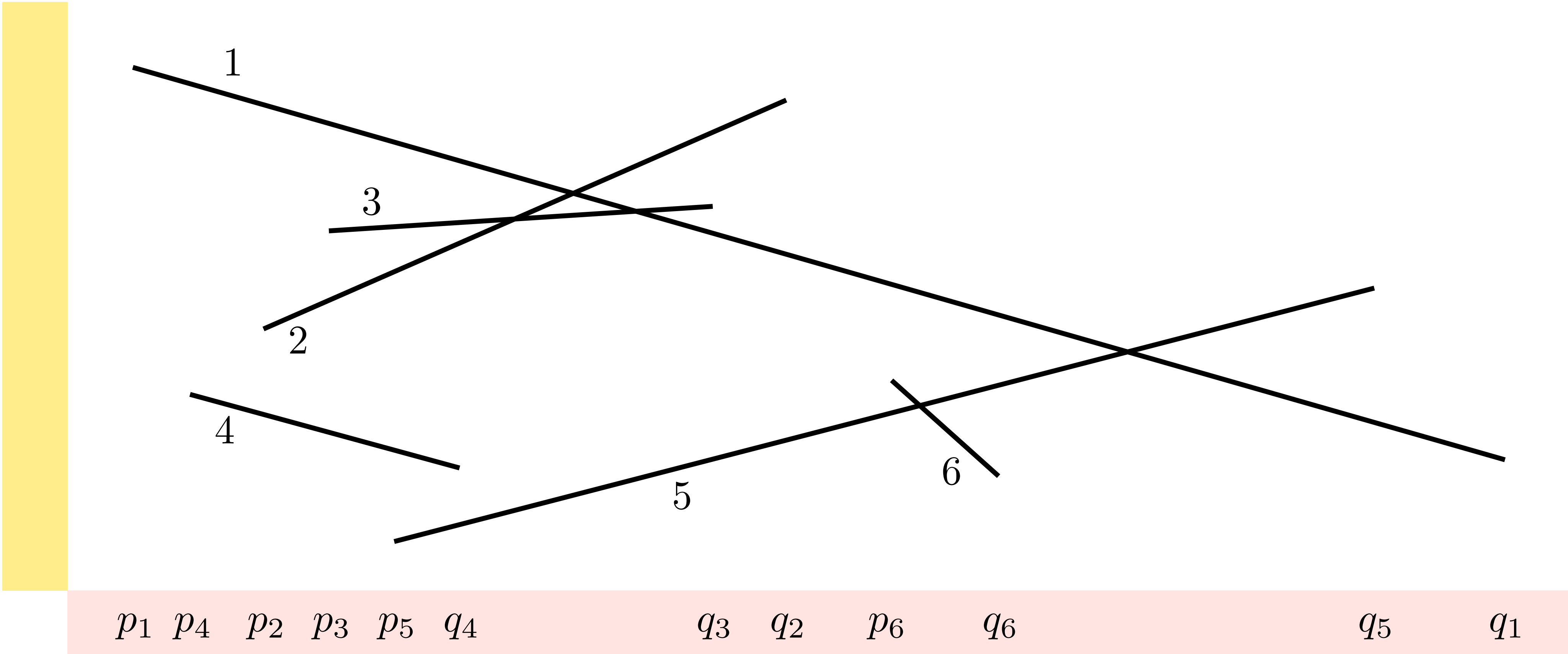
Bentley-Ottman's algorithm: simulation



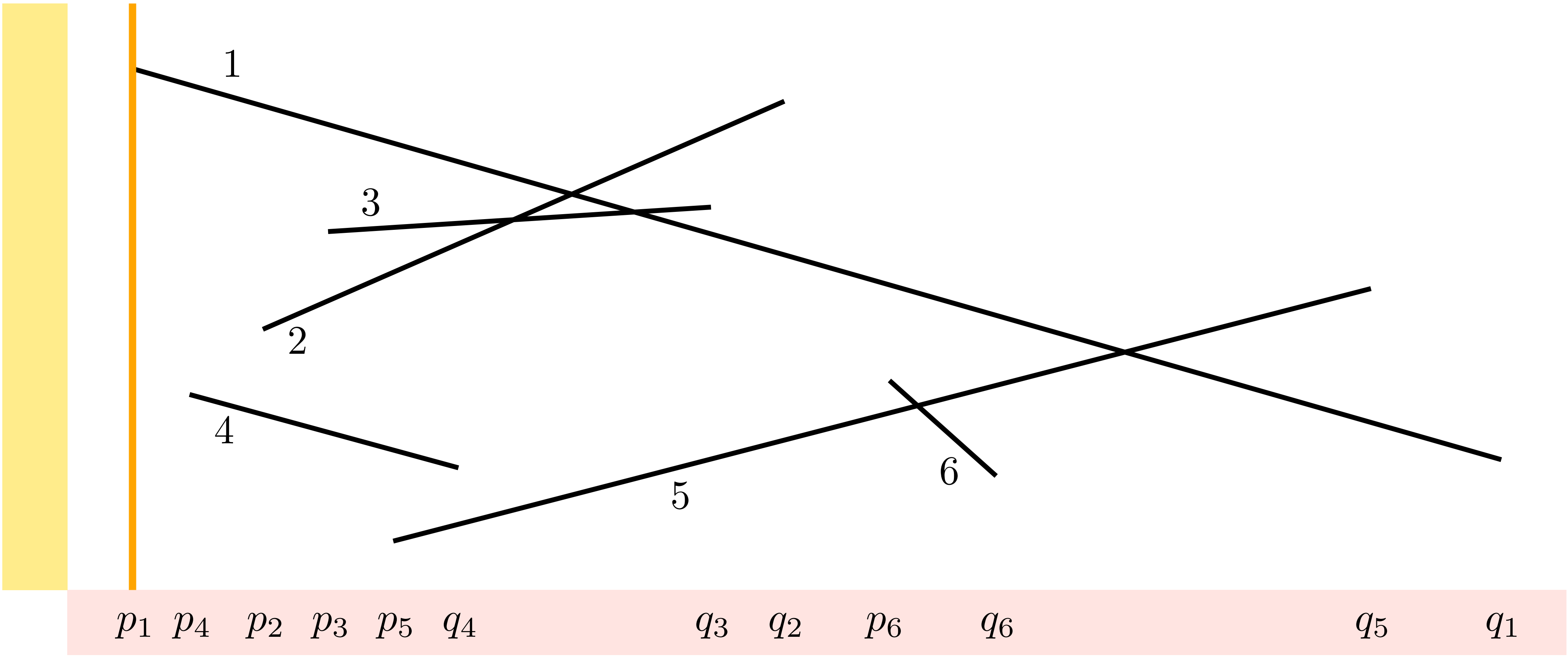
Bentley-Ottman's algorithm: simulation



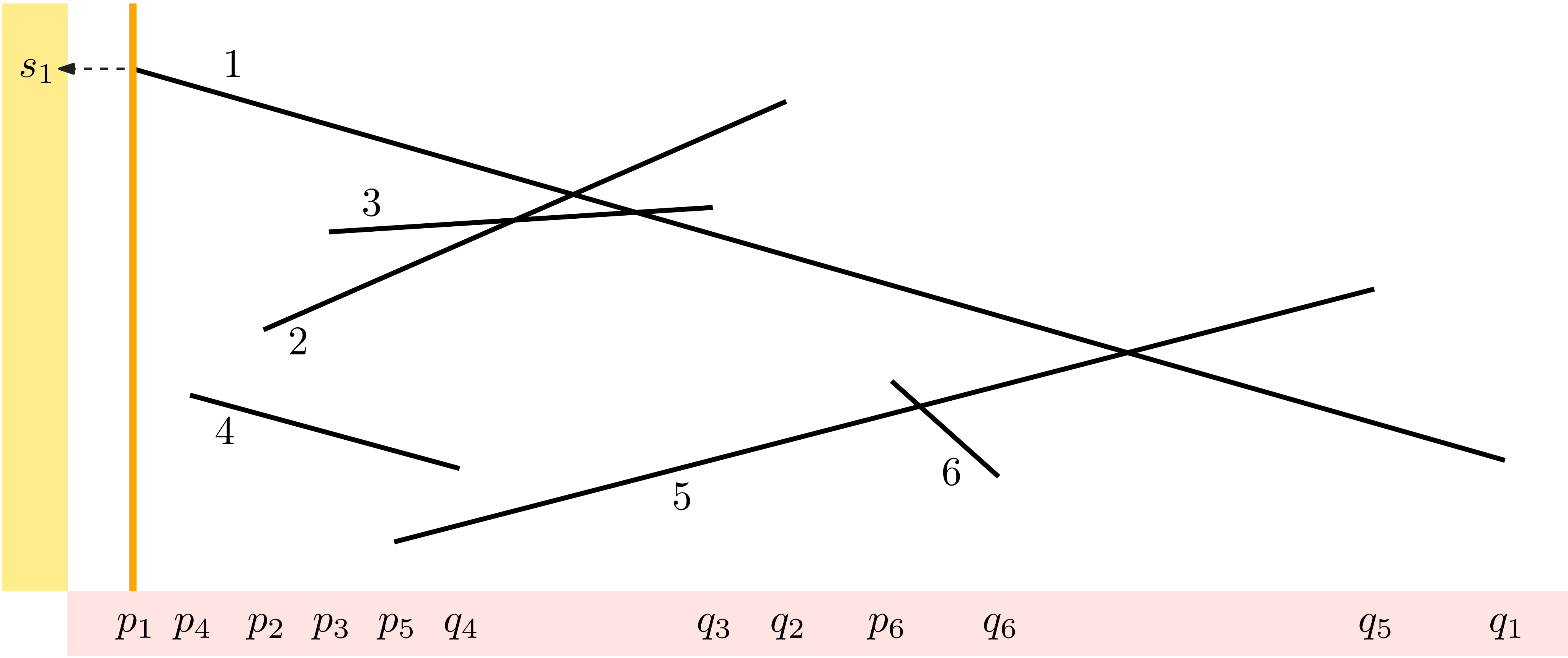
Bentley-Ottman's algorithm: simulation



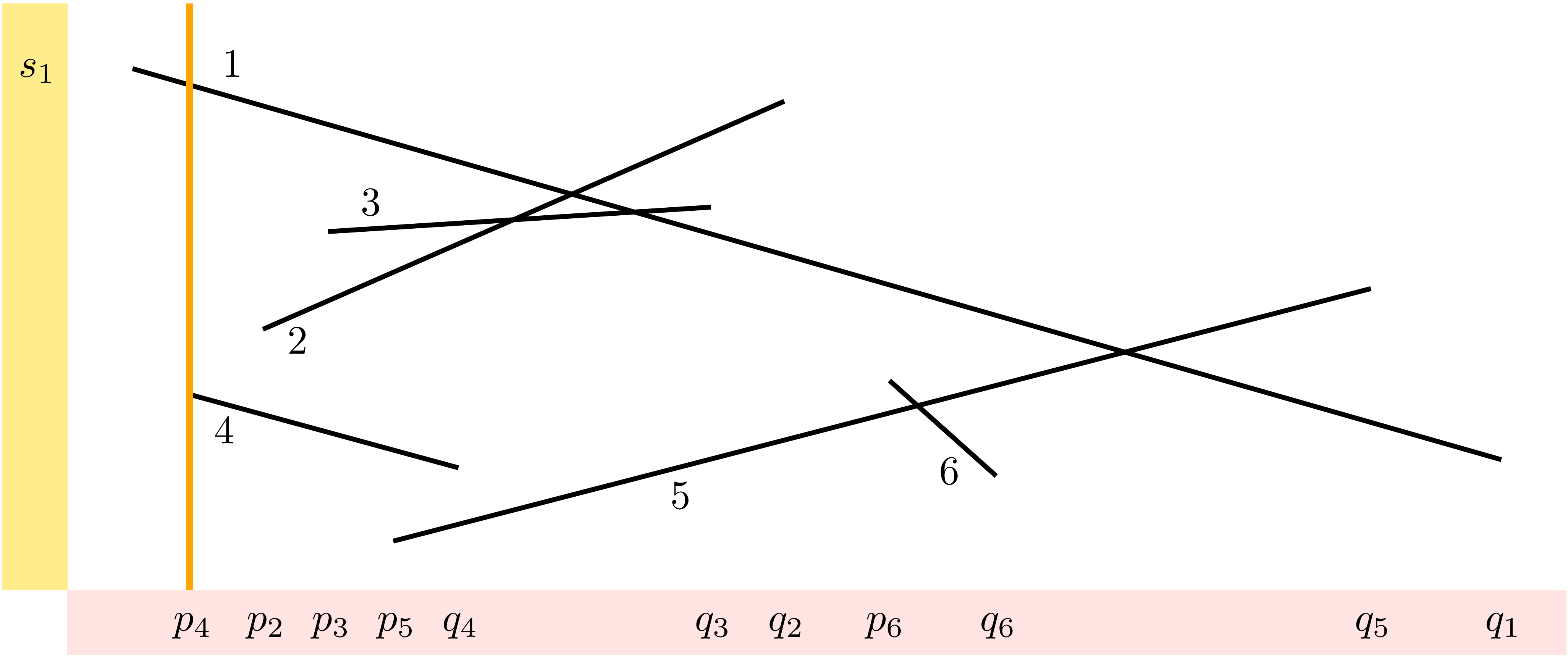
Bentley-Ottman's algorithm: simulation



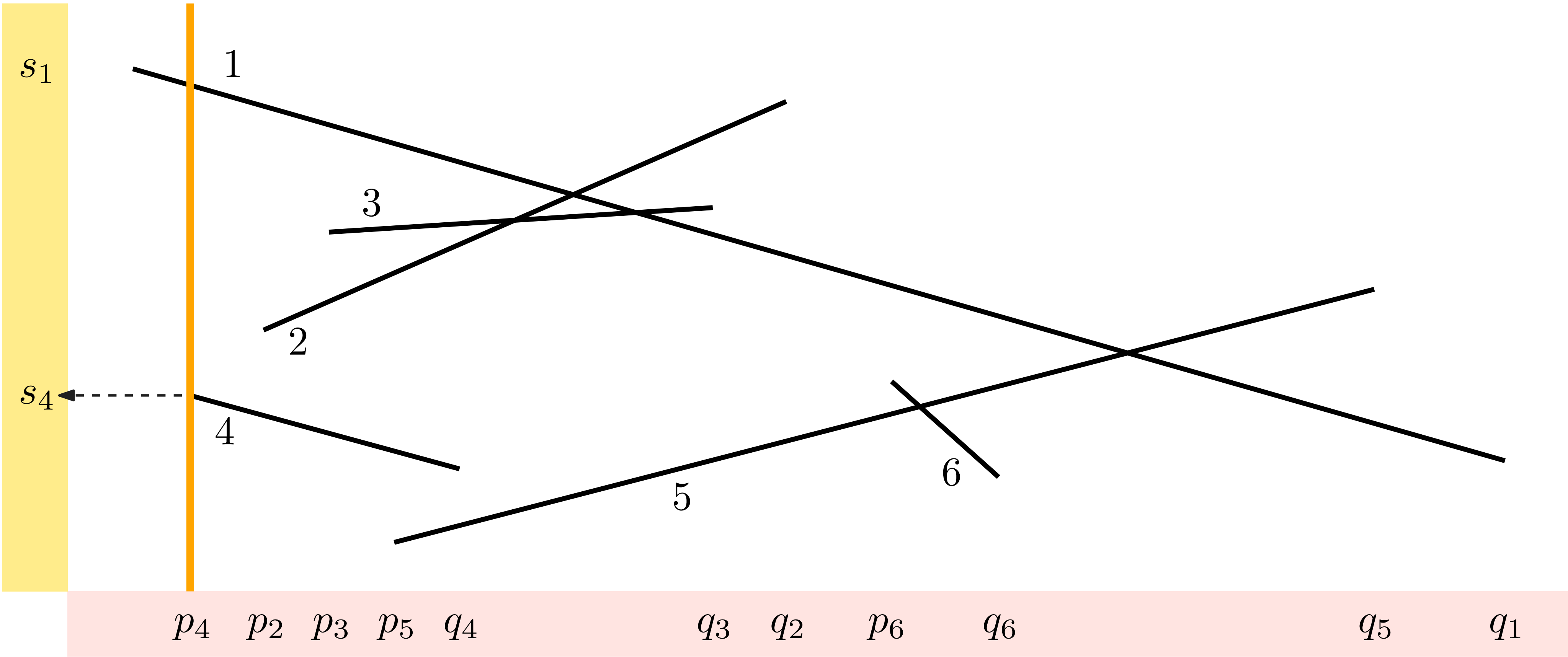
Bentley-Ottman's algorithm: simulation



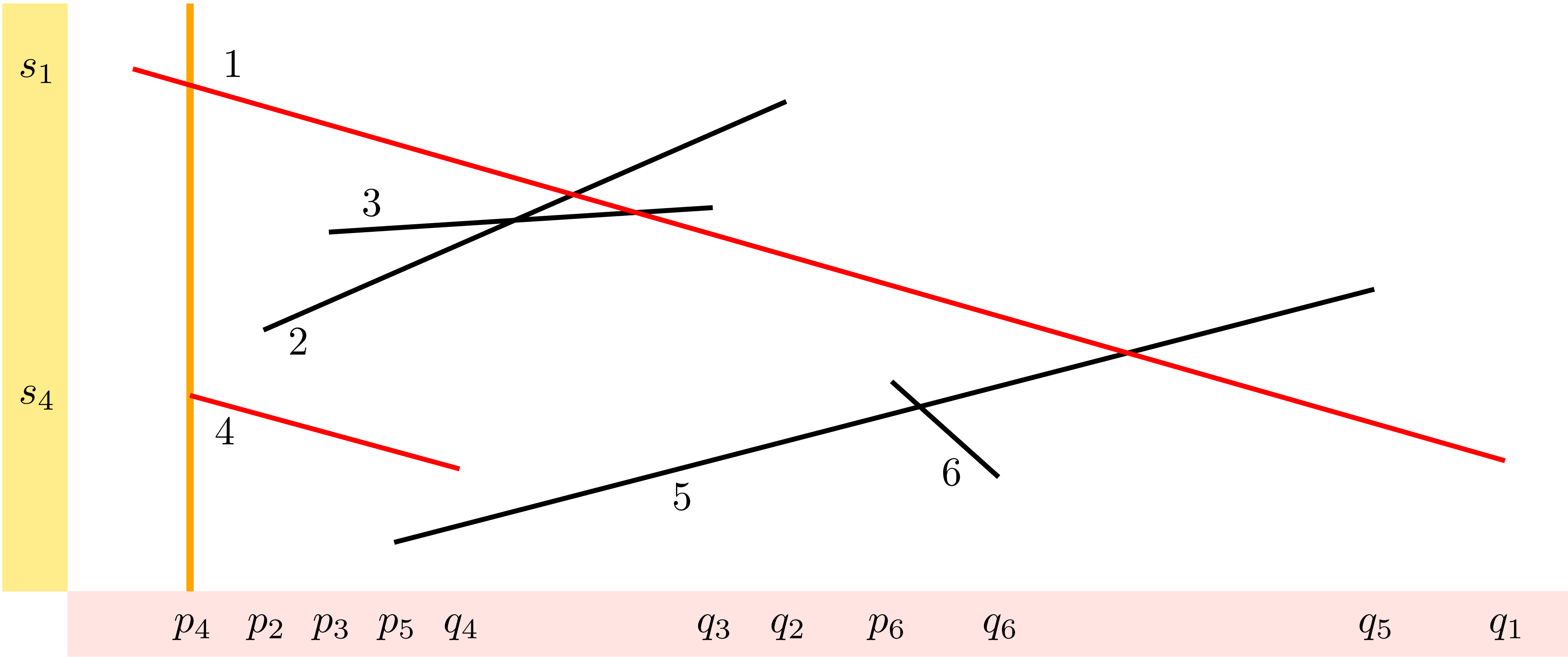
Bentley-Ottman's algorithm: simulation



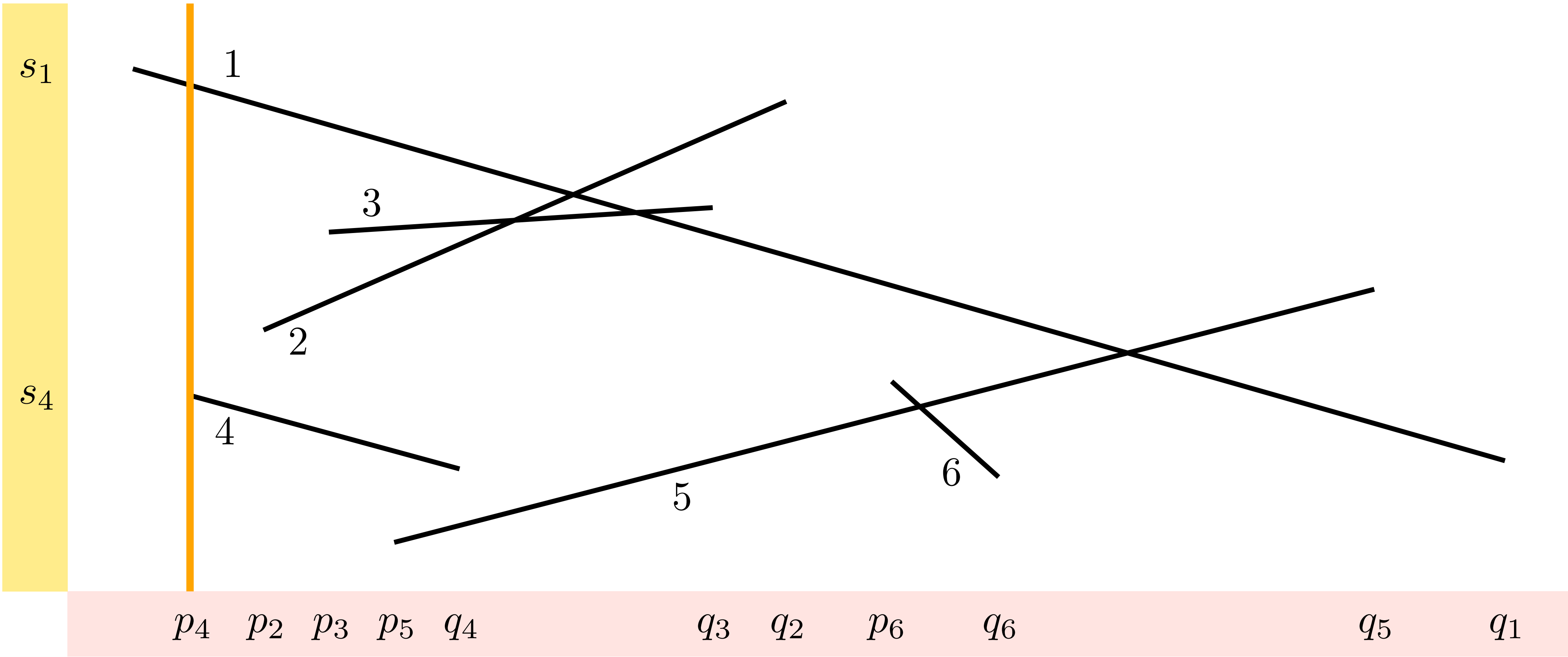
Bentley-Ottman's algorithm: simulation



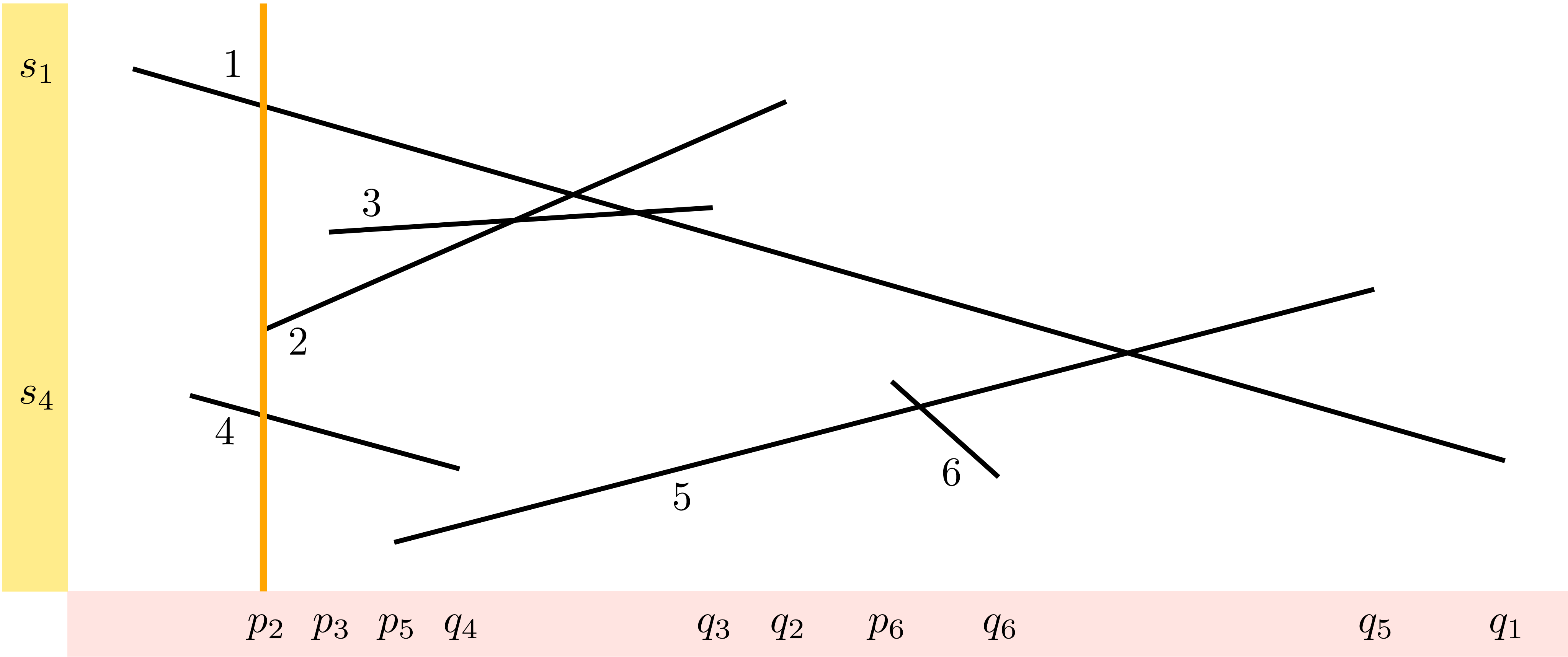
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

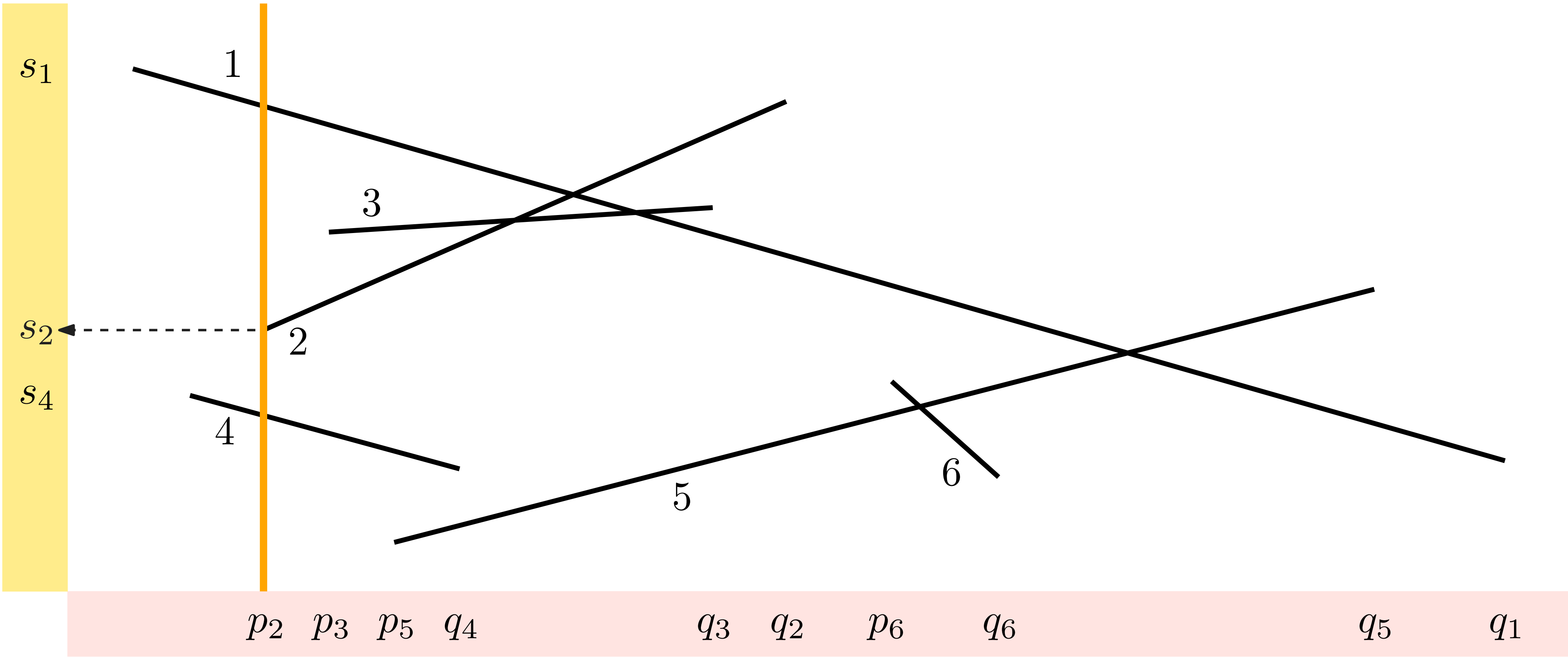


Bentley-Ottman's algorithm: simulation

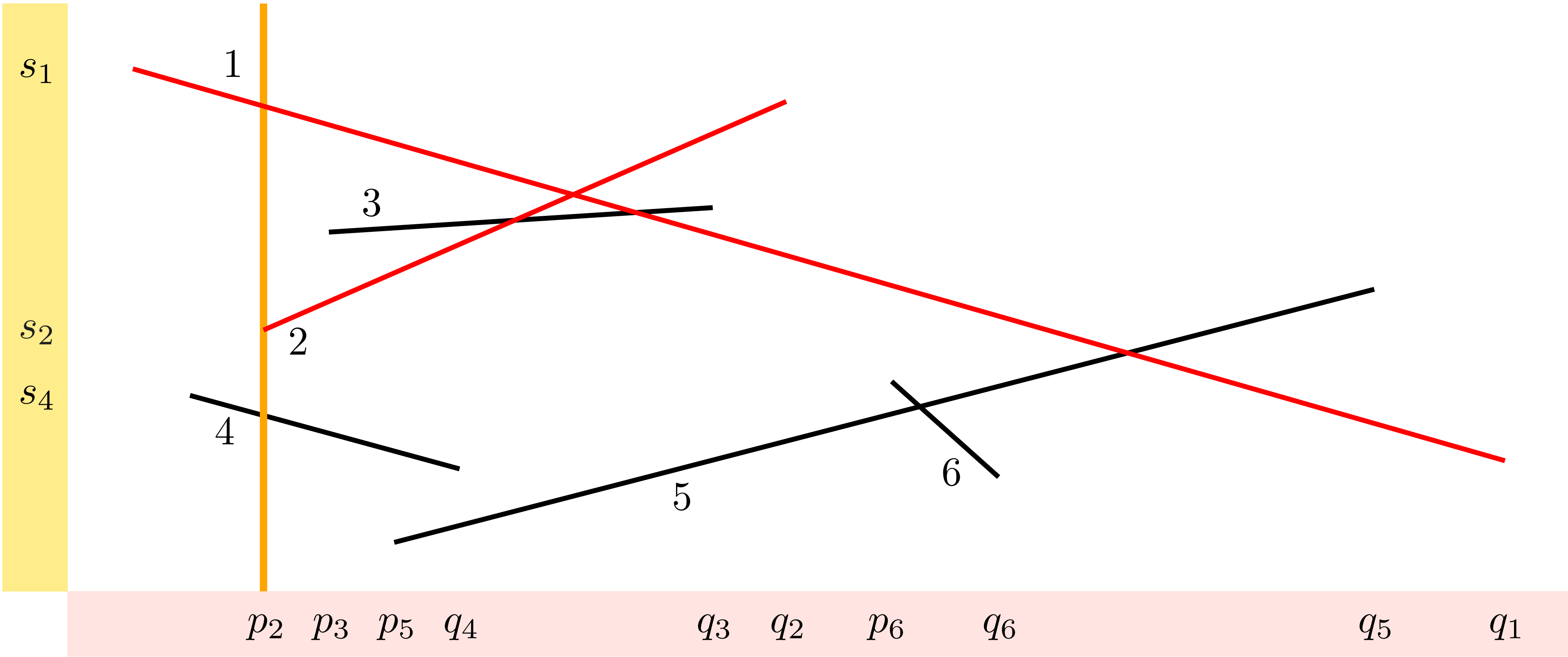


INTERSECTION OF LINE-SEGMENTS

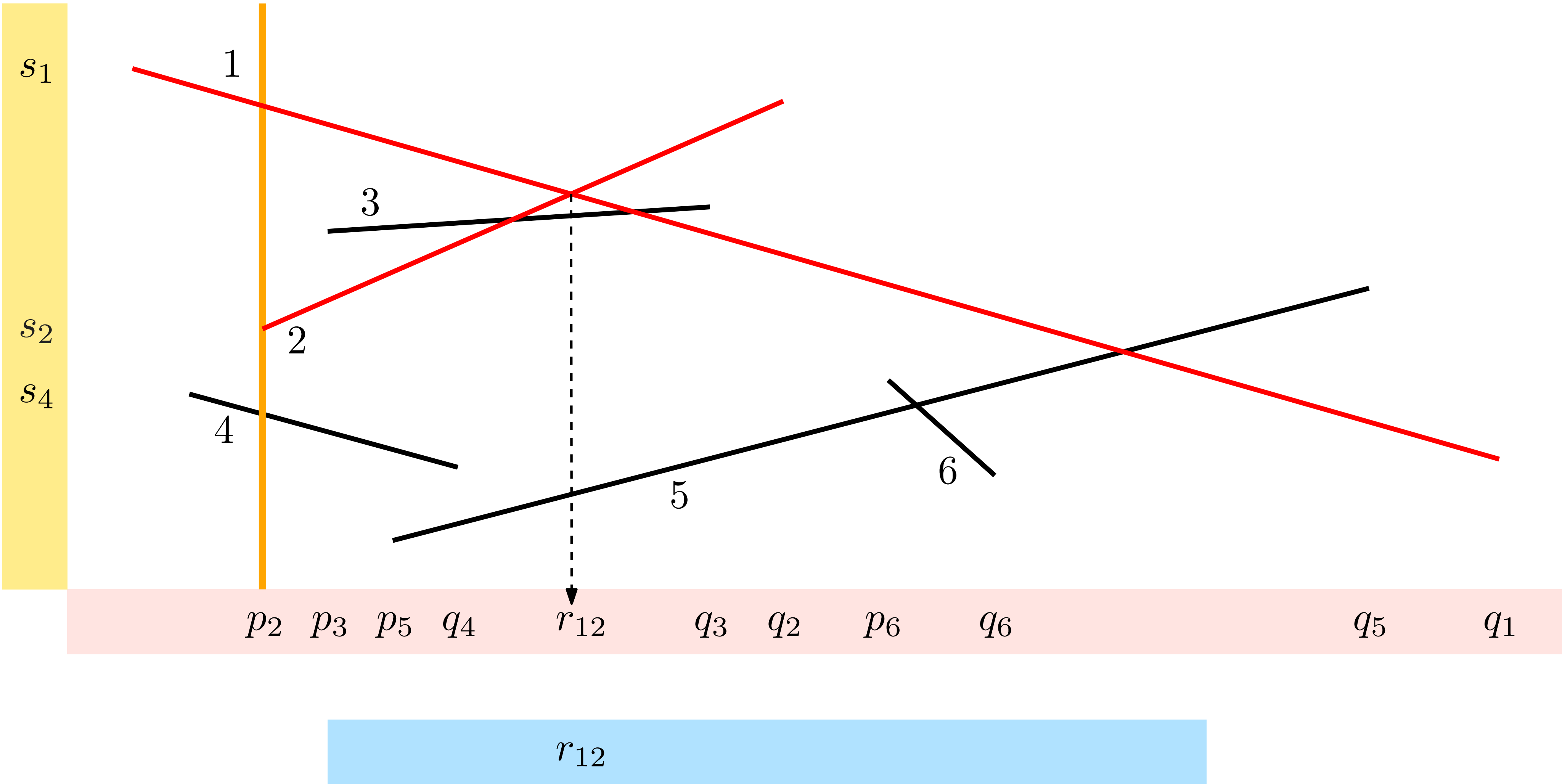
Bentley-Ottman's algorithm: simulation



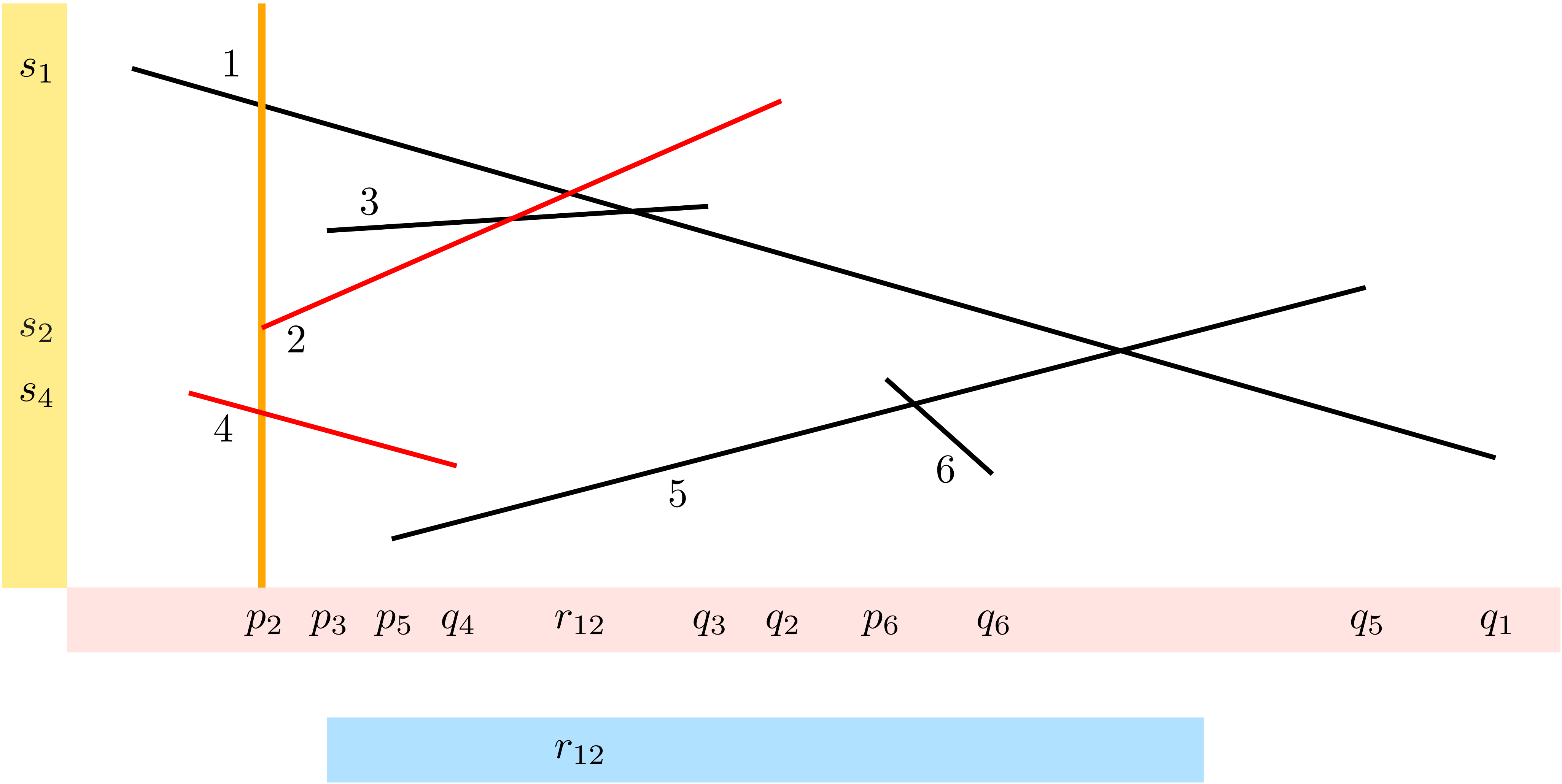
Bentley-Ottman's algorithm: simulation



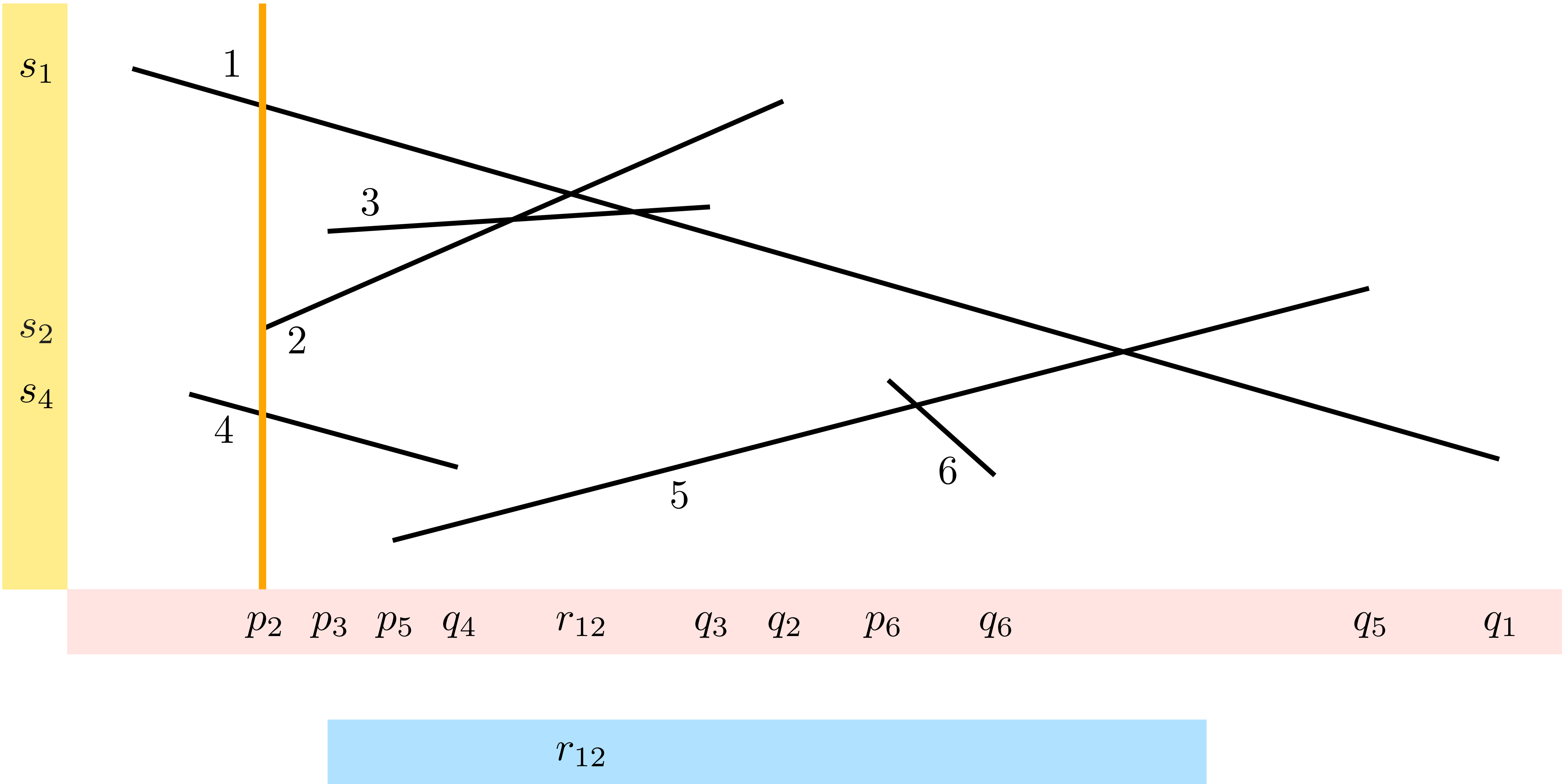
Bentley-Ottman's algorithm: simulation



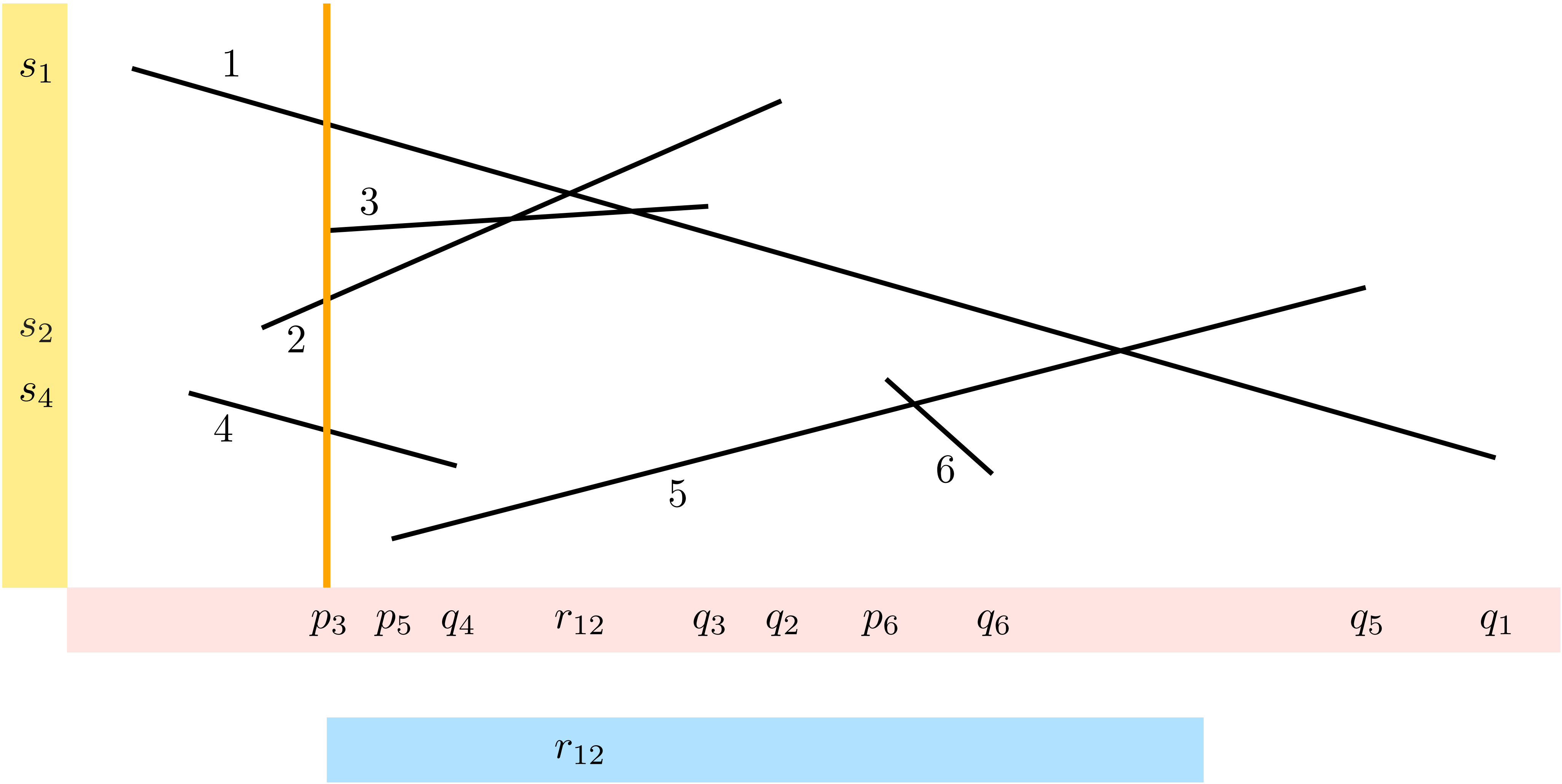
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

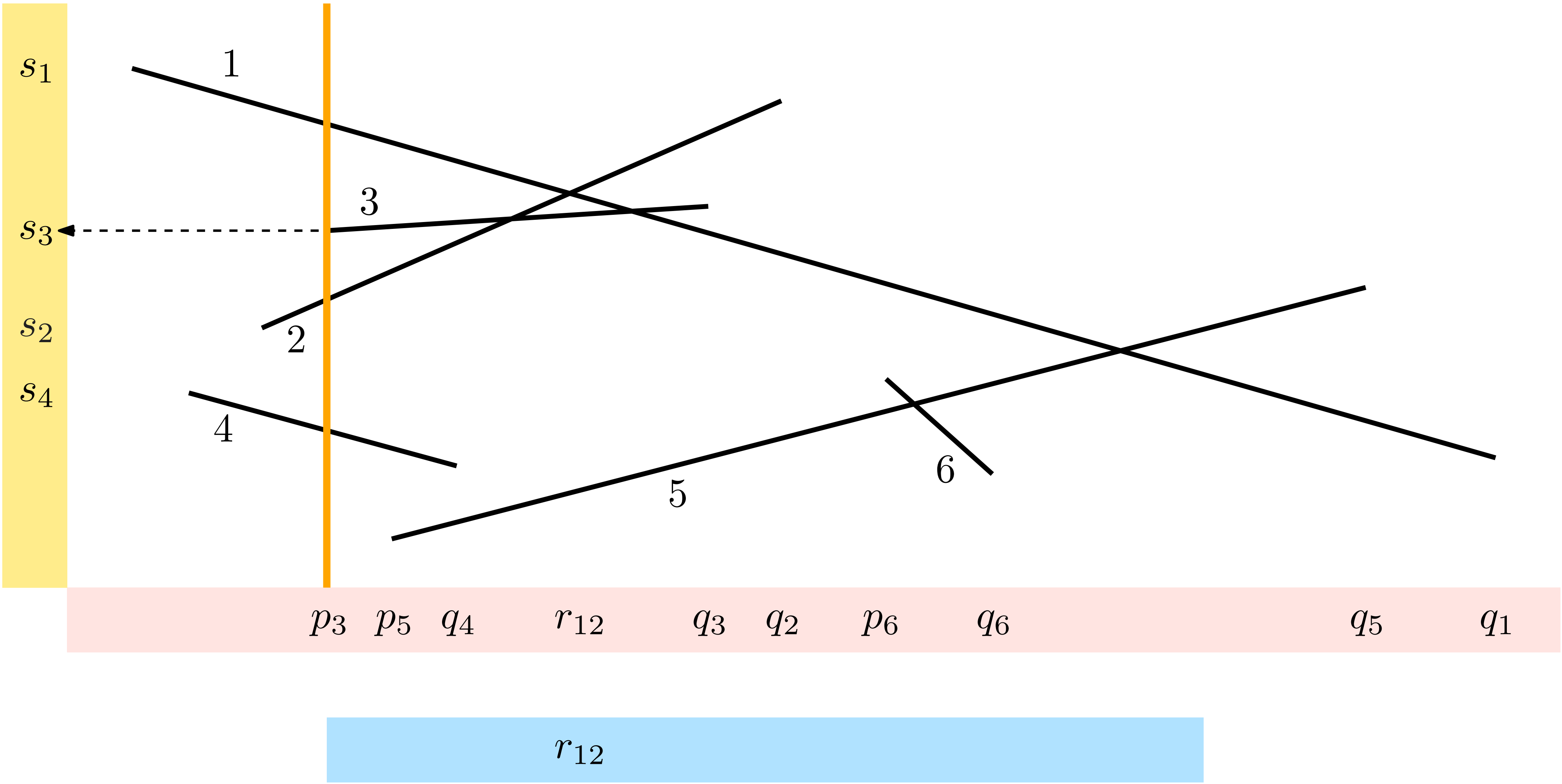


Bentley-Ottman's algorithm: simulation

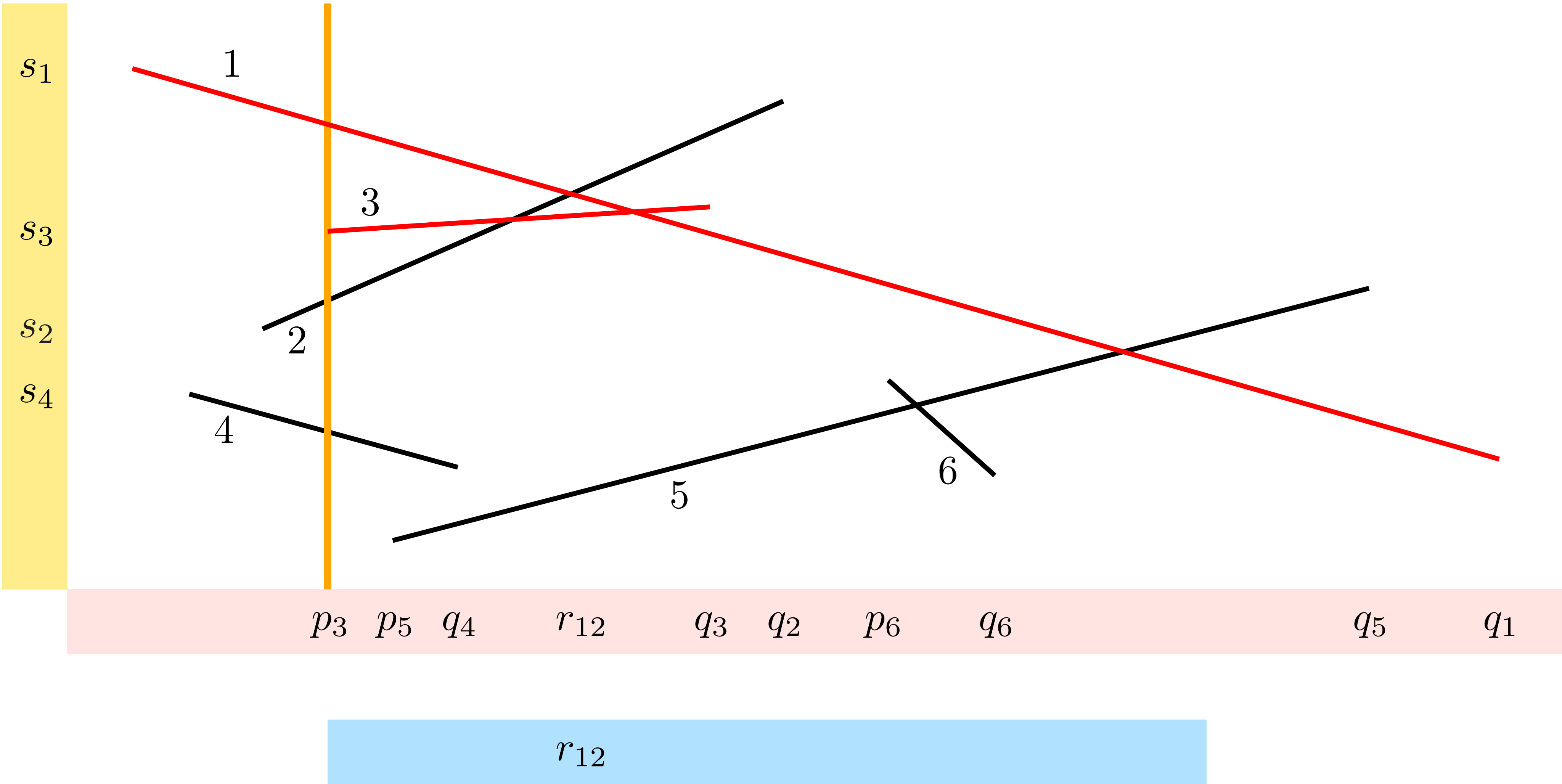


INTERSECTION OF LINE-SEGMENTS

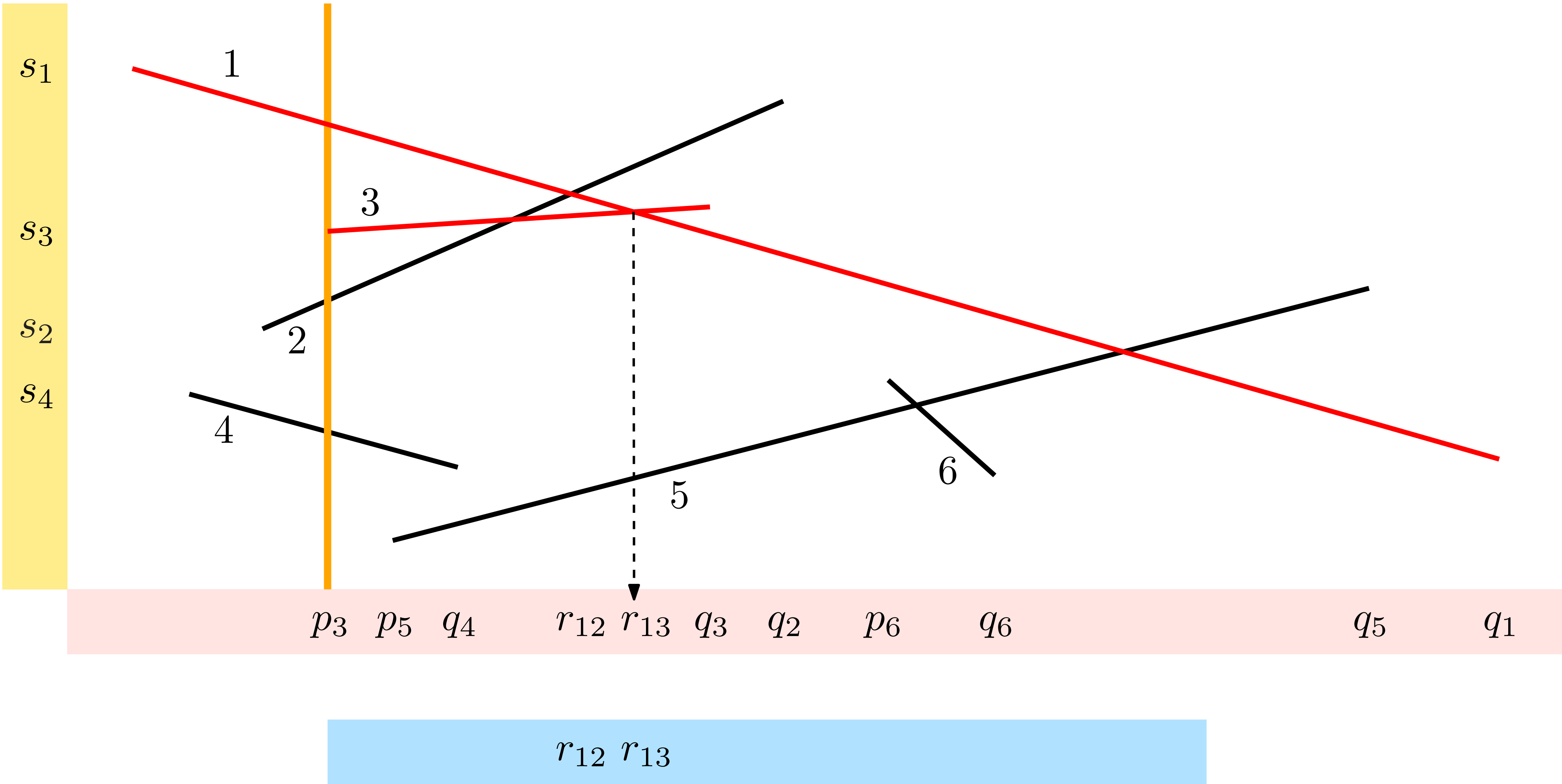
Bentley-Ottman's algorithm: simulation



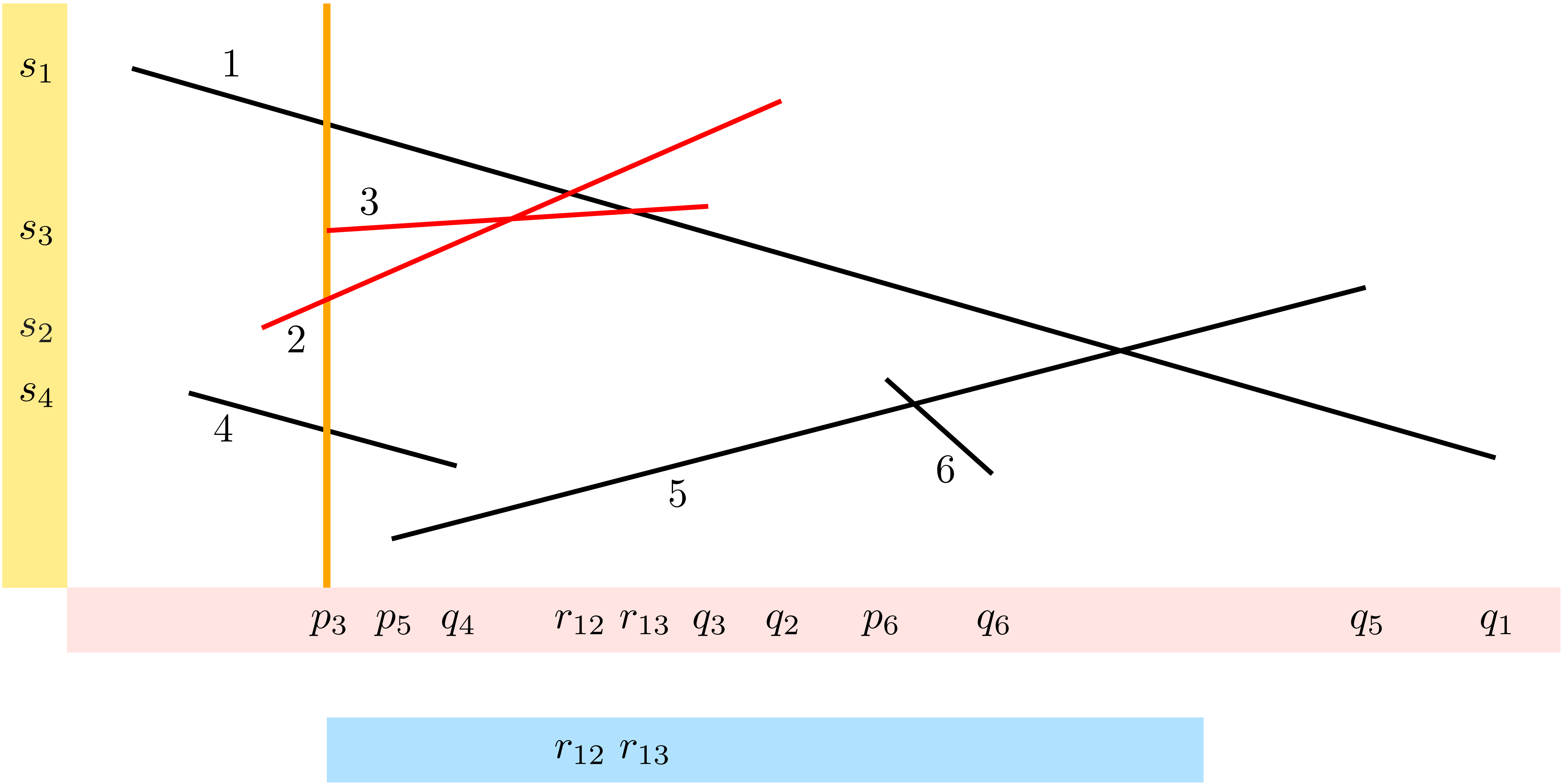
Bentley-Ottman's algorithm: simulation



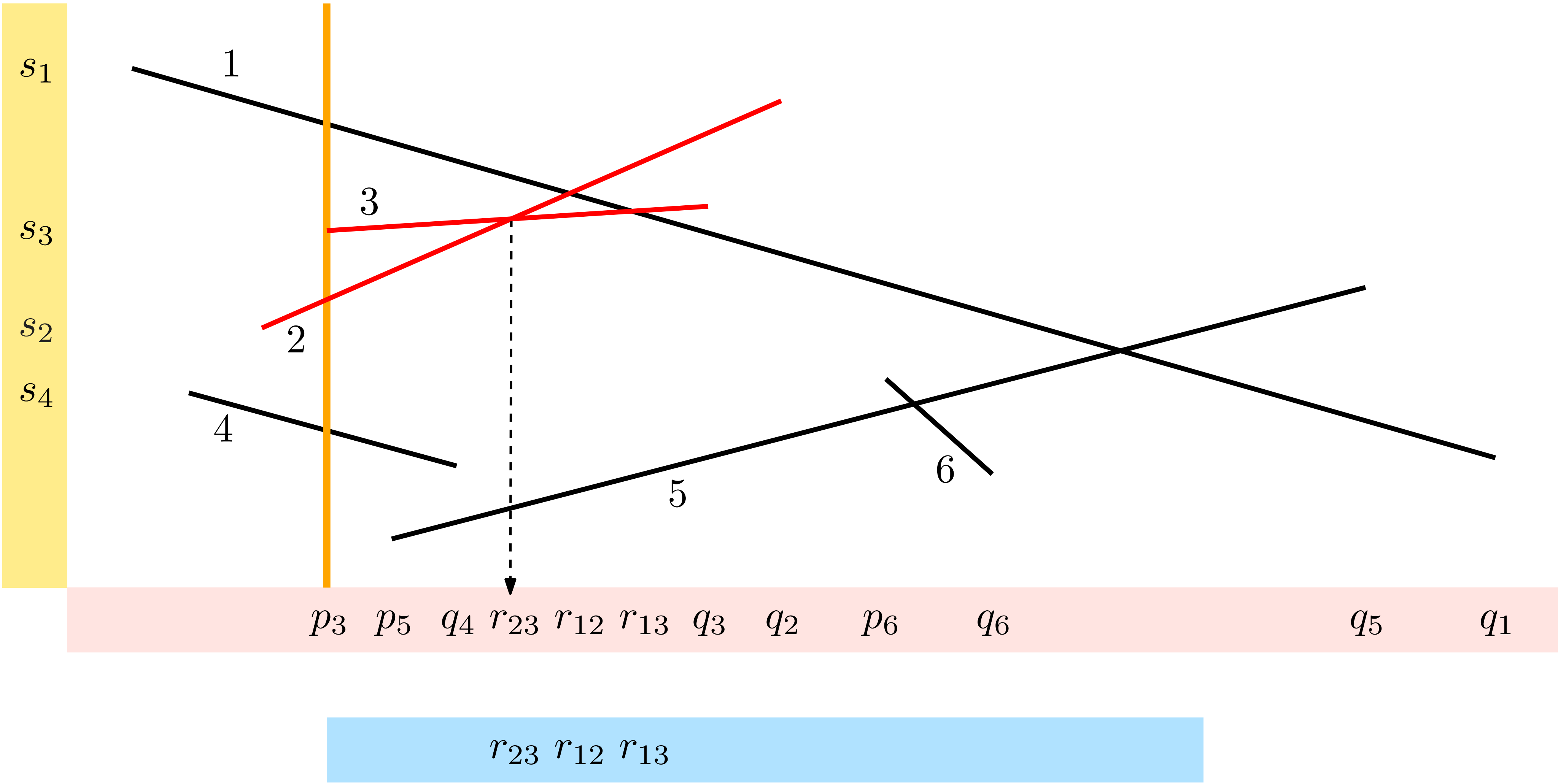
Bentley-Ottman's algorithm: simulation



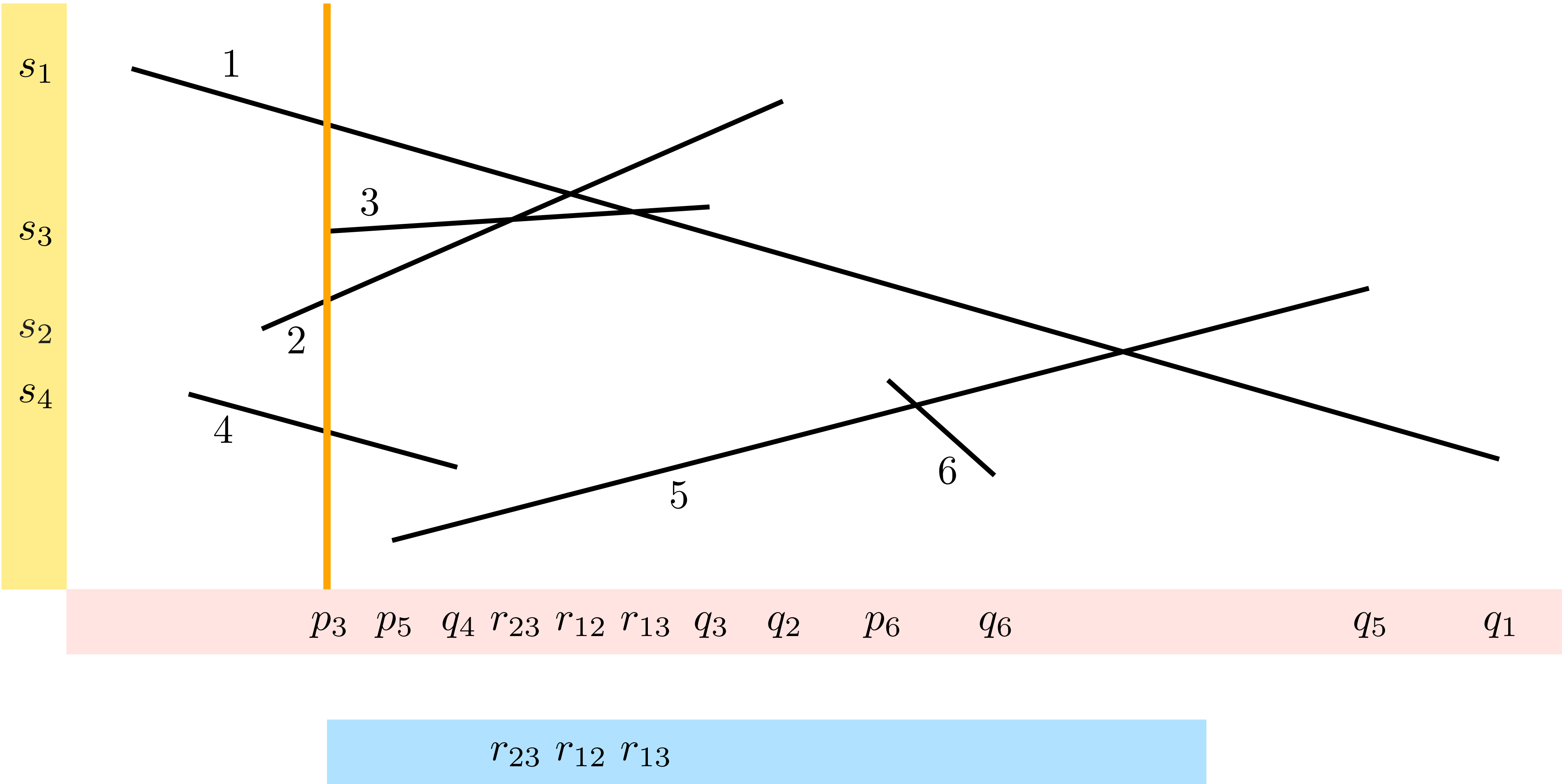
Bentley-Ottman's algorithm: simulation



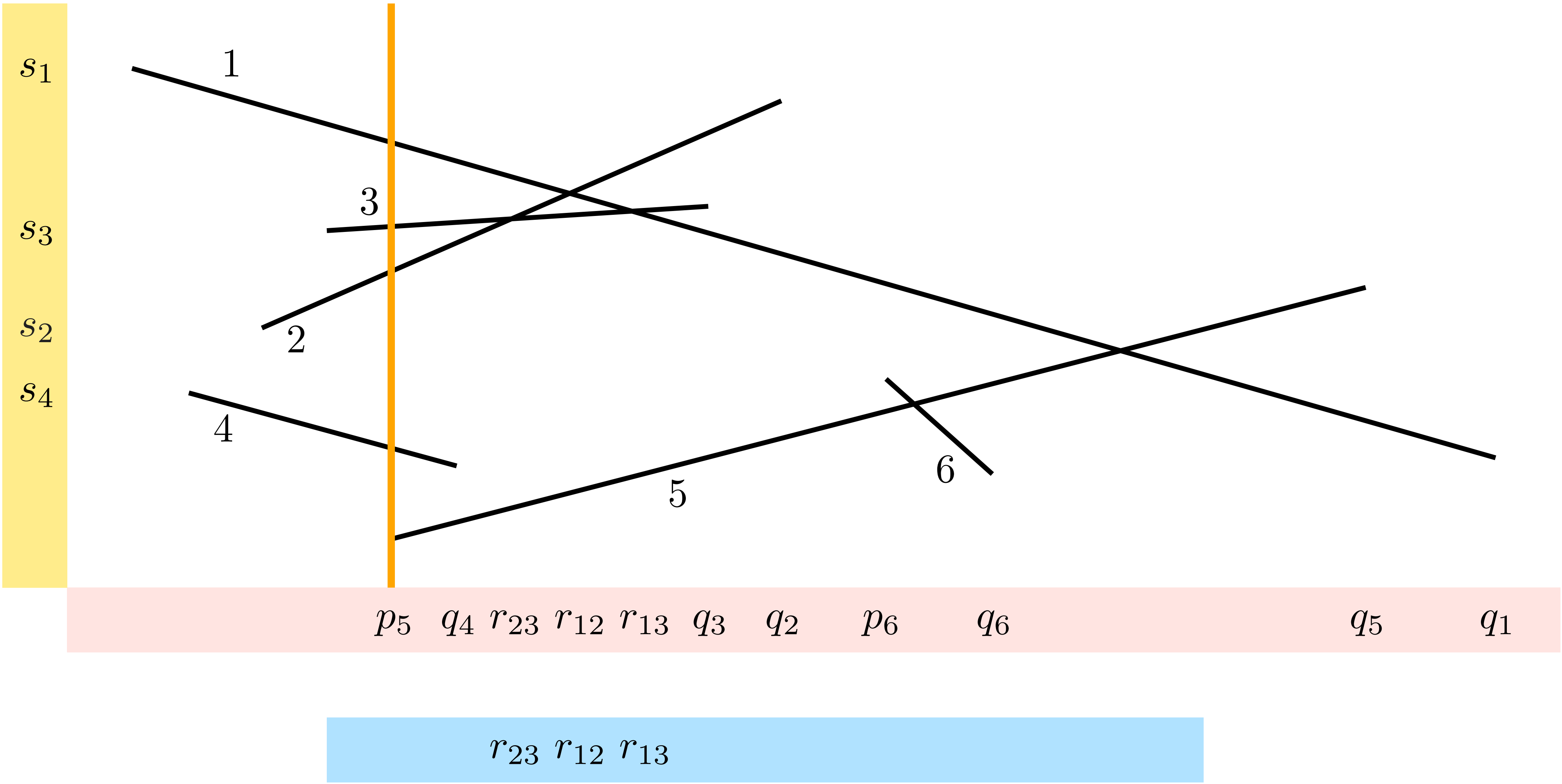
Bentley-Ottman's algorithm: simulation



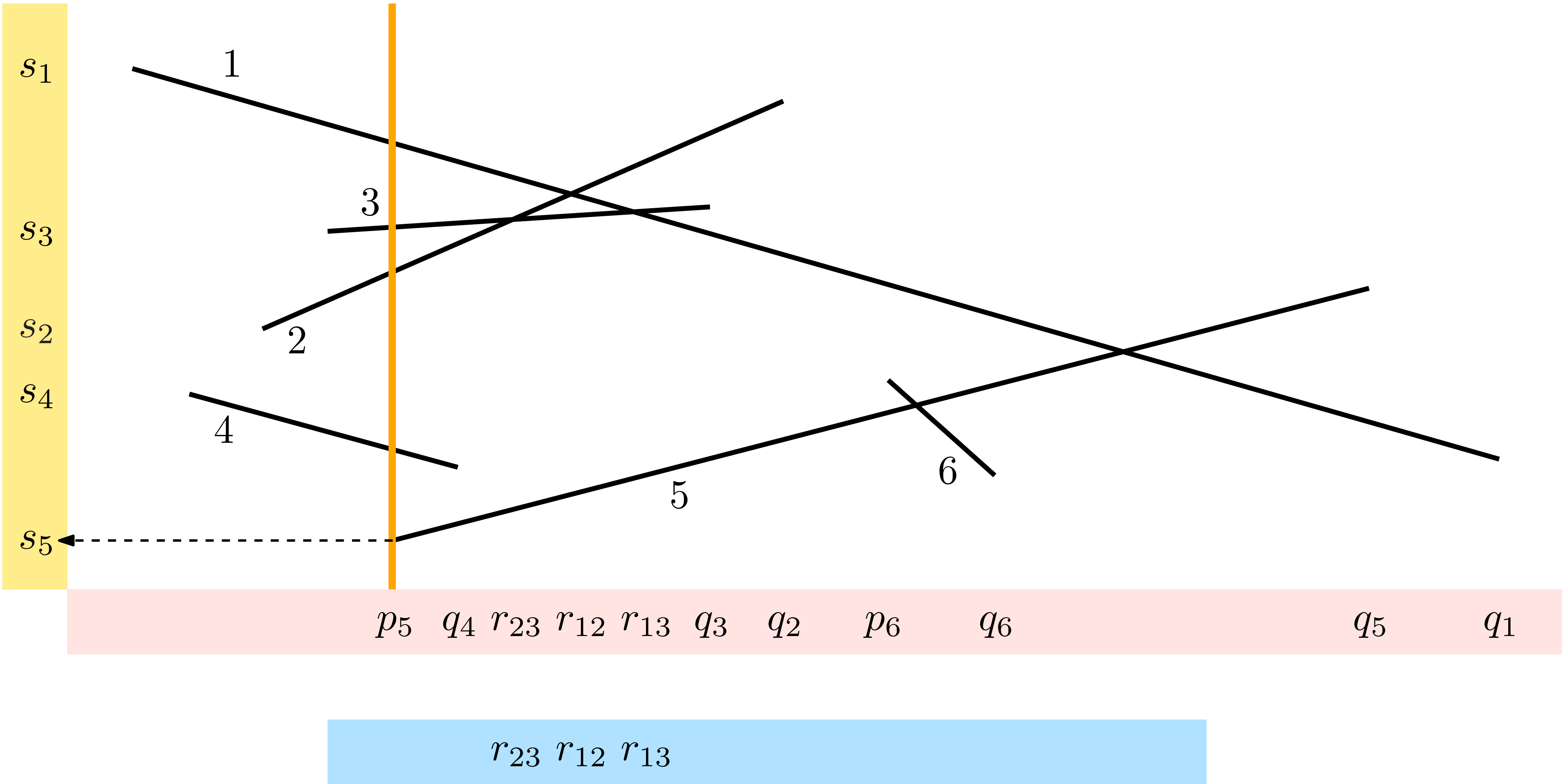
Bentley-Ottman's algorithm: simulation



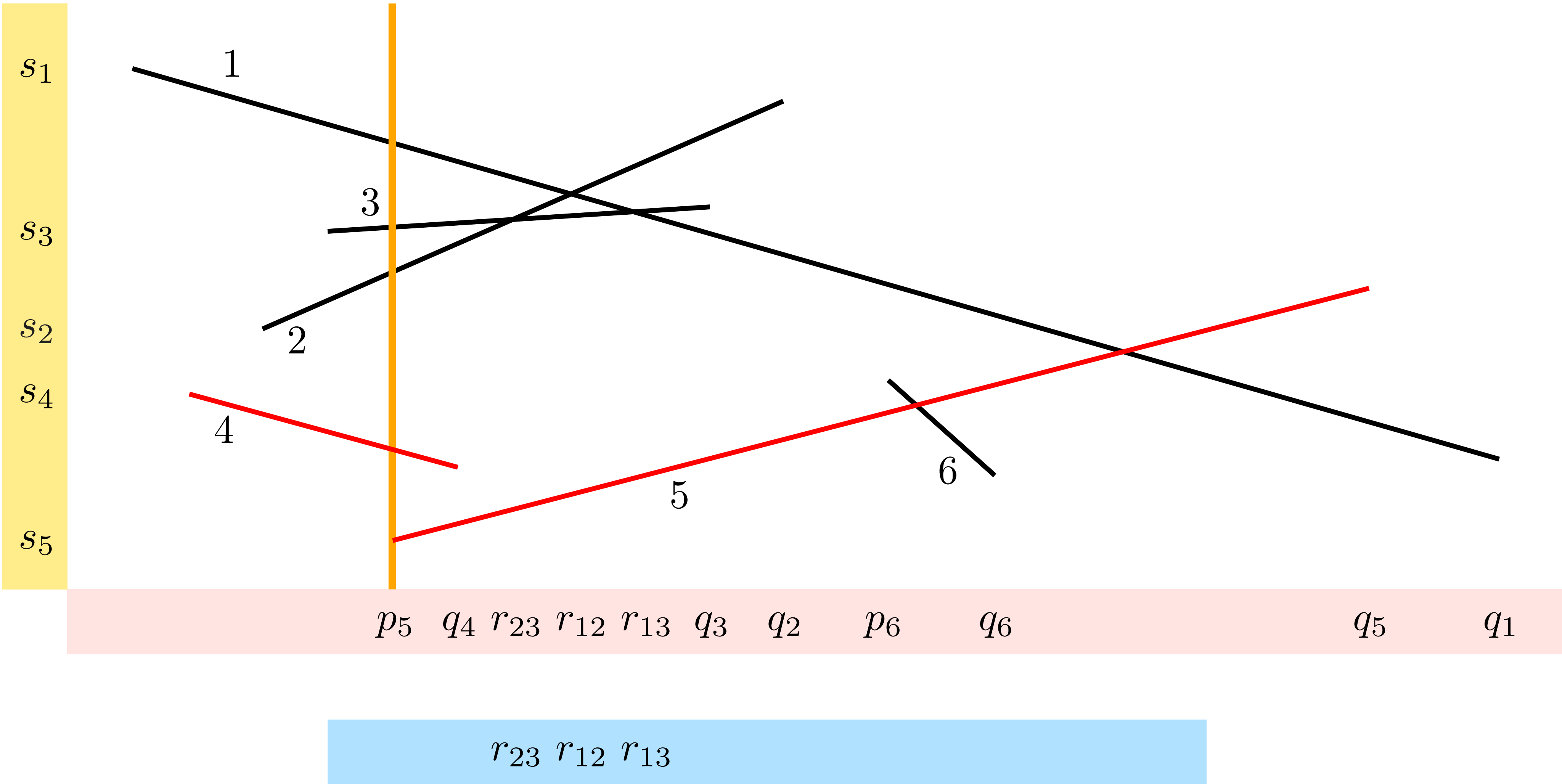
Bentley-Ottman's algorithm: simulation



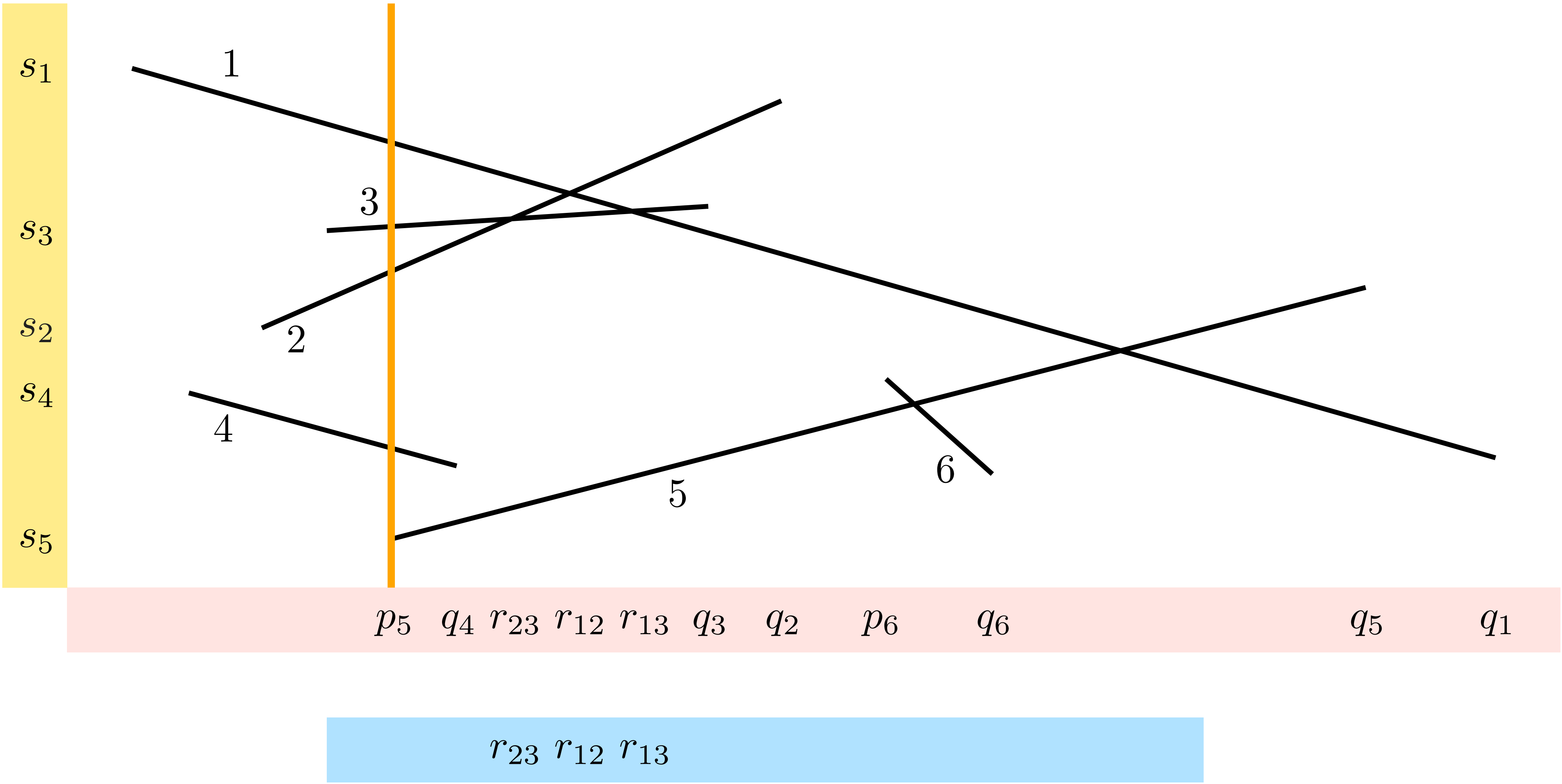
Bentley-Ottman's algorithm: simulation



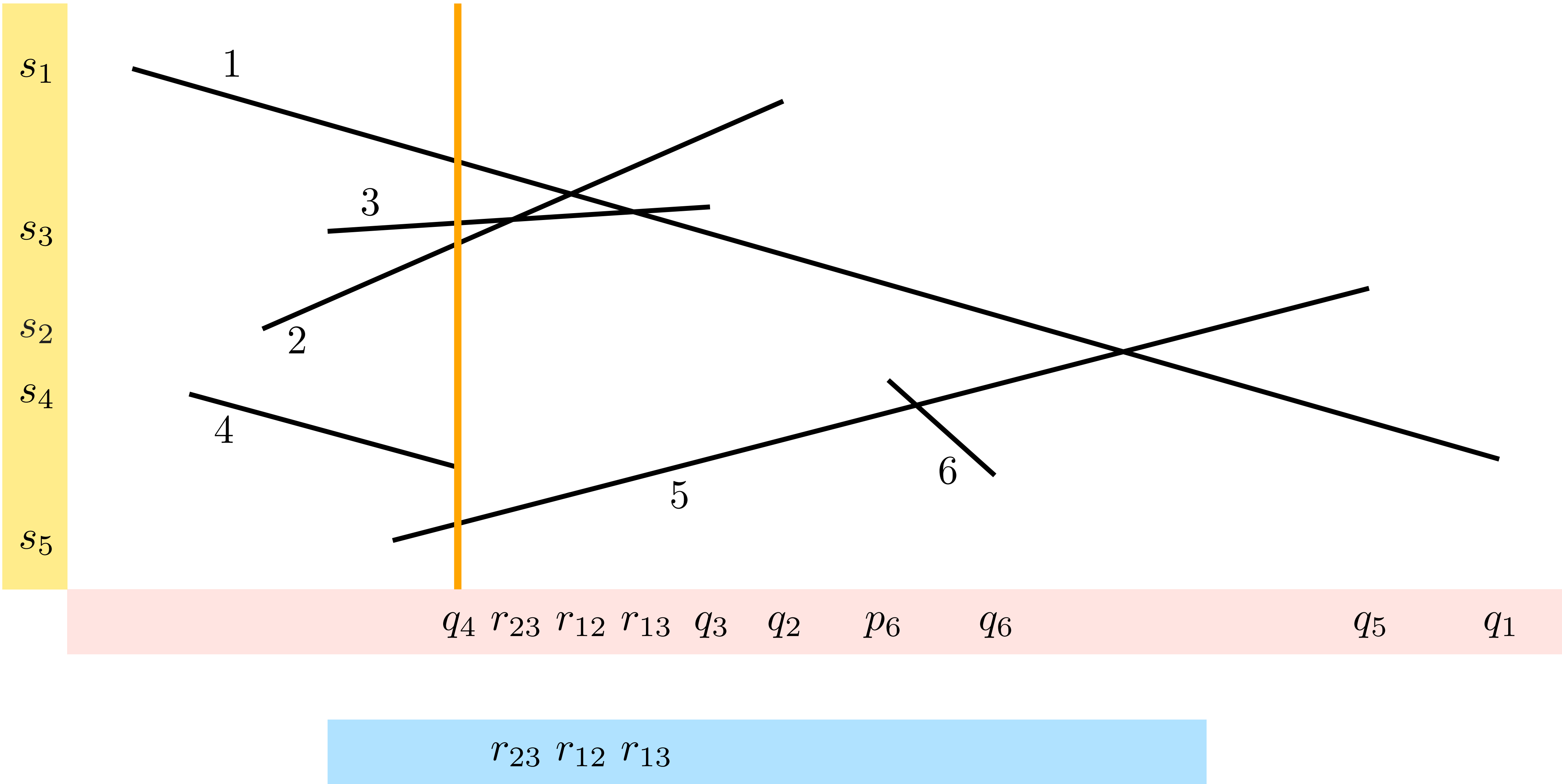
Bentley-Ottman's algorithm: simulation



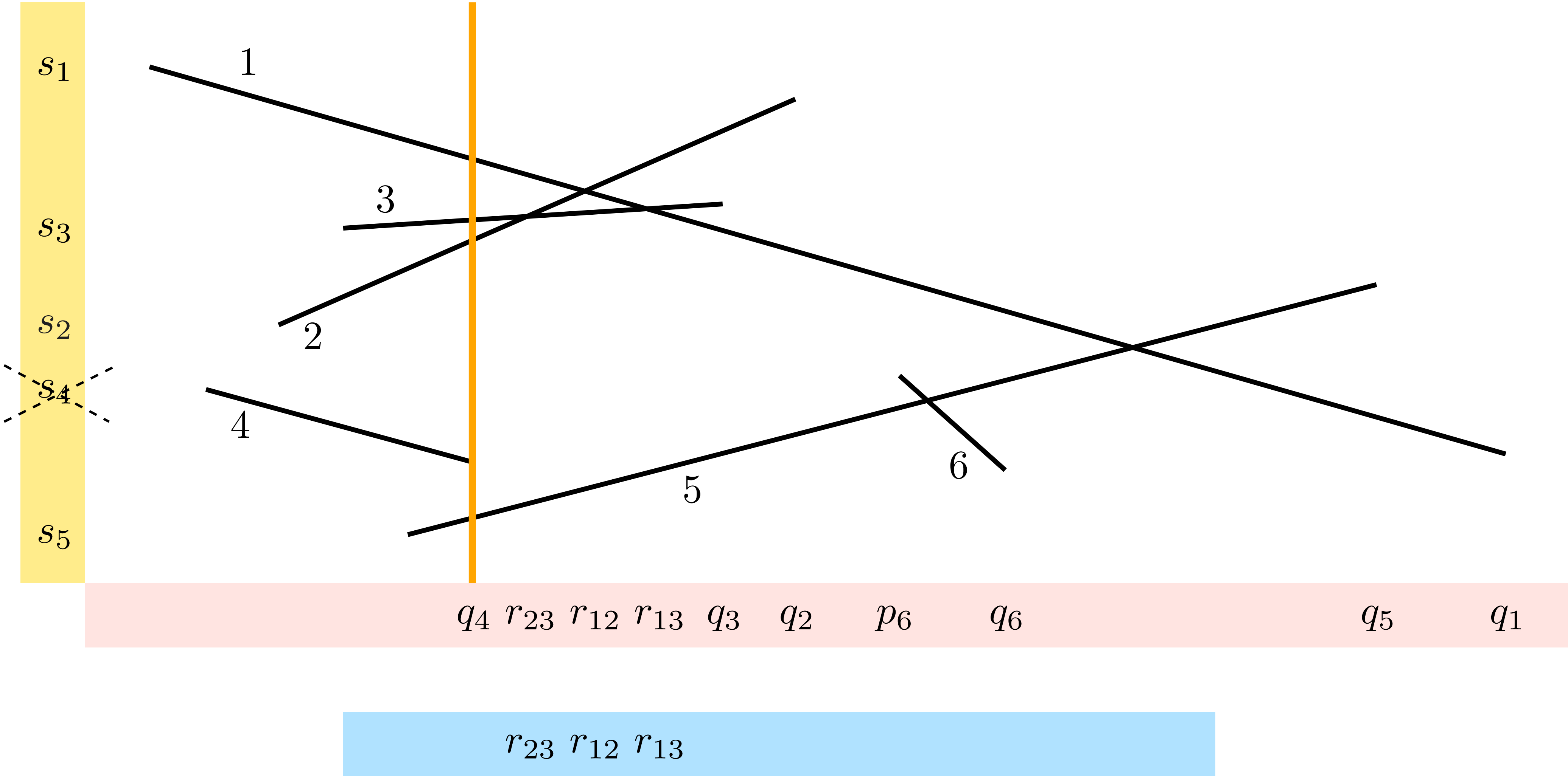
Bentley-Ottman's algorithm: simulation



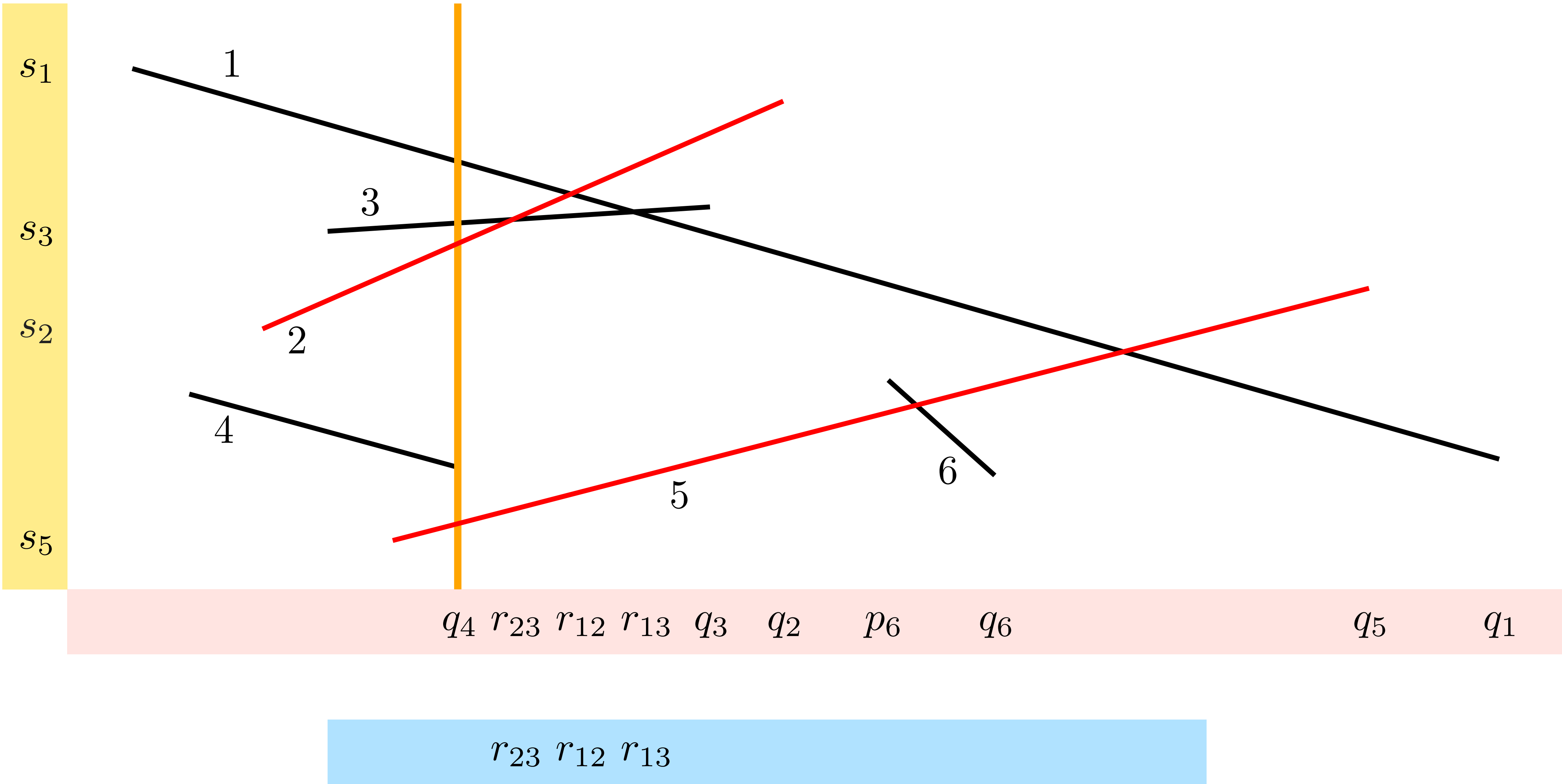
Bentley-Ottman's algorithm: simulation



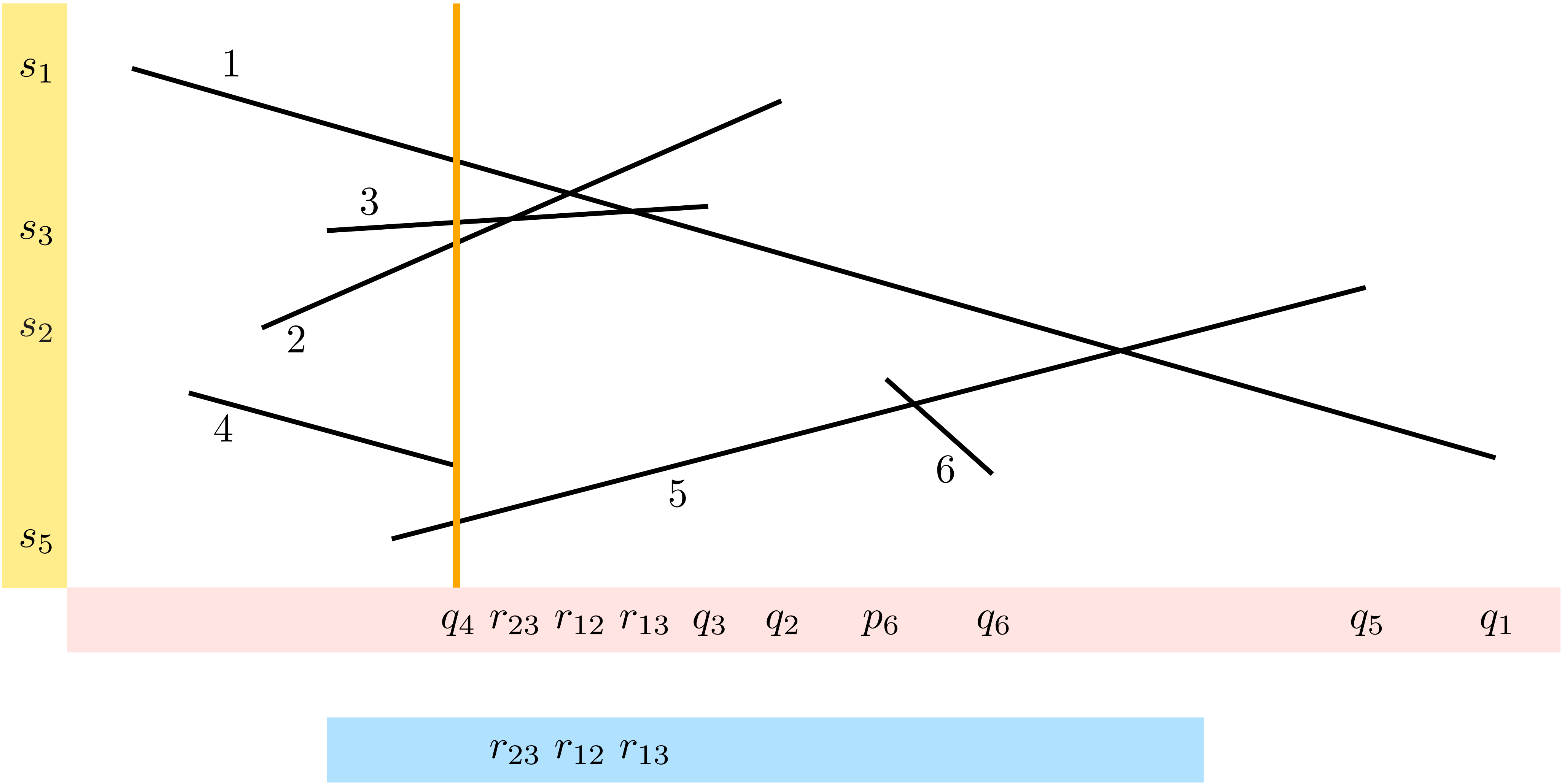
Bentley-Ottman's algorithm: simulation



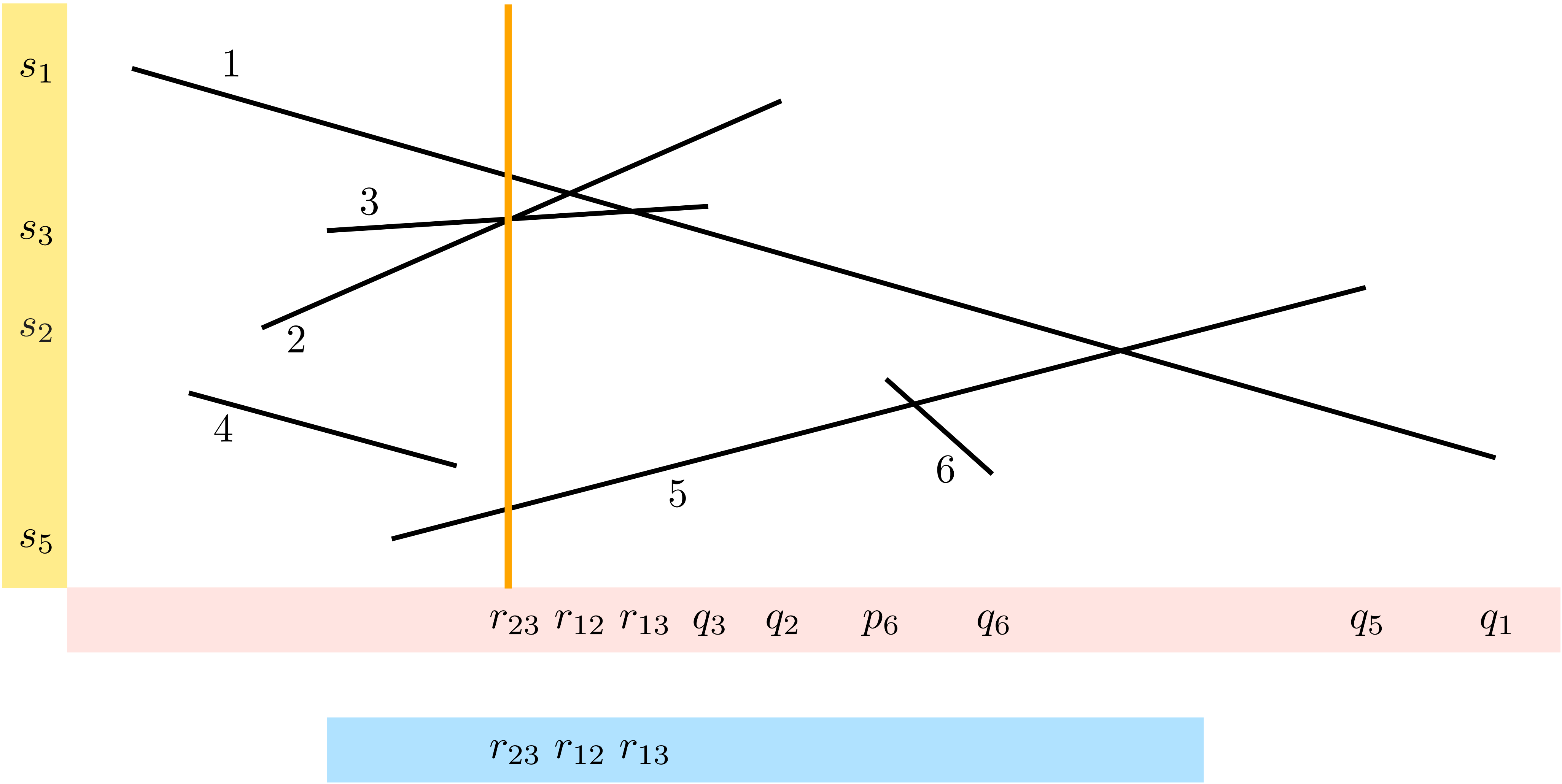
Bentley-Ottman's algorithm: simulation



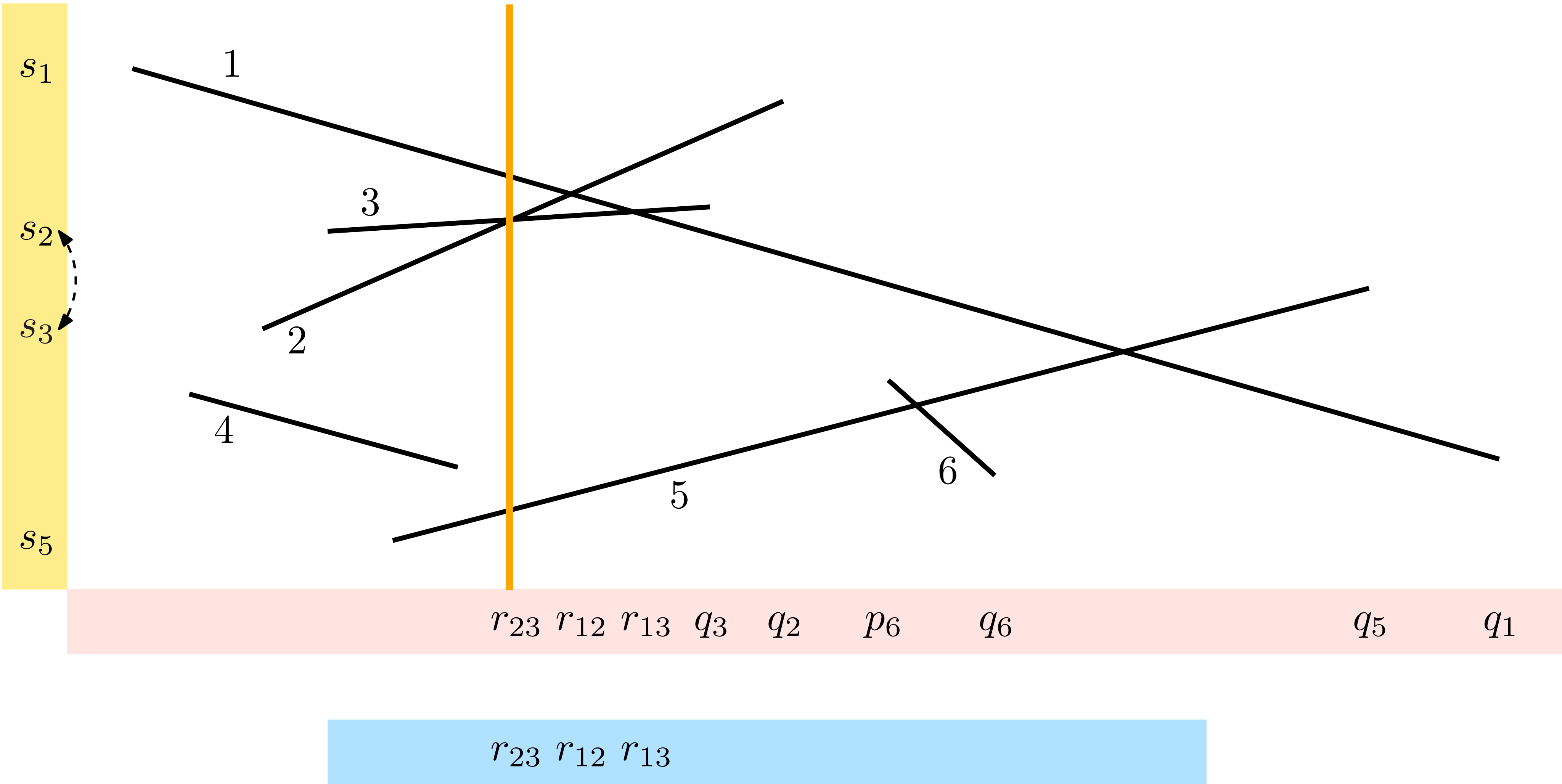
Bentley-Ottman's algorithm: simulation



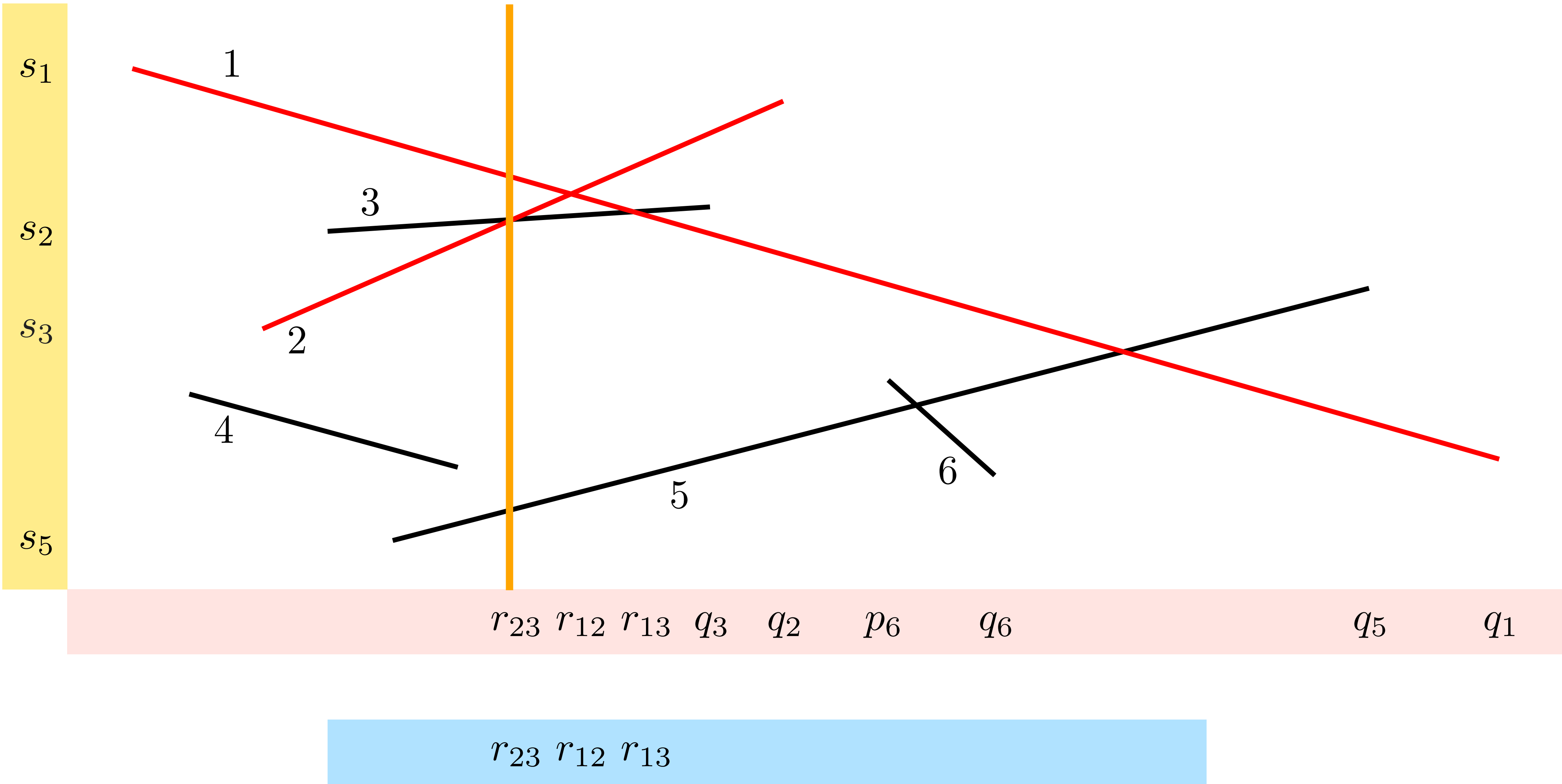
Bentley-Ottman's algorithm: simulation



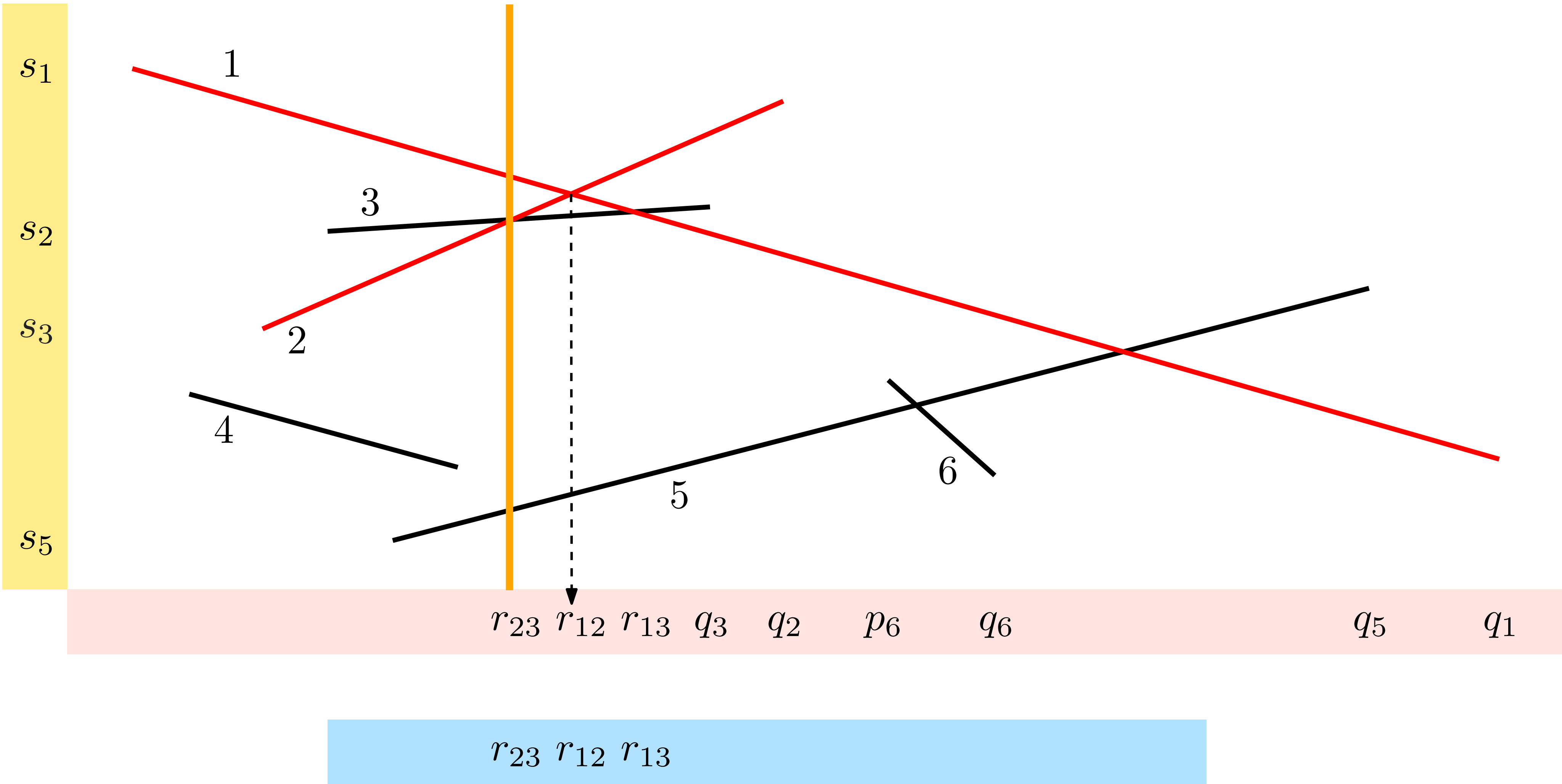
Bentley-Ottman's algorithm: simulation



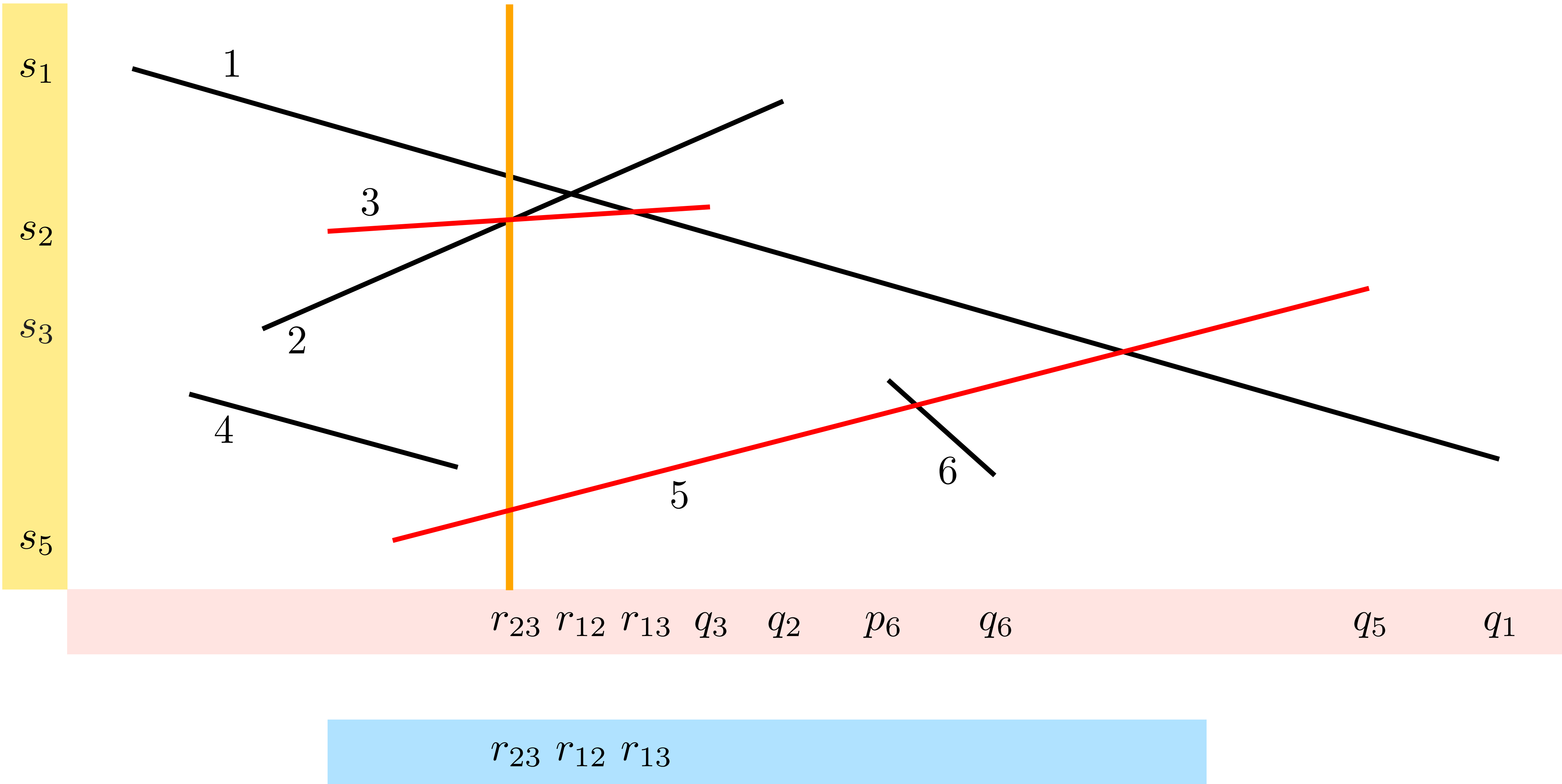
Bentley-Ottman's algorithm: simulation



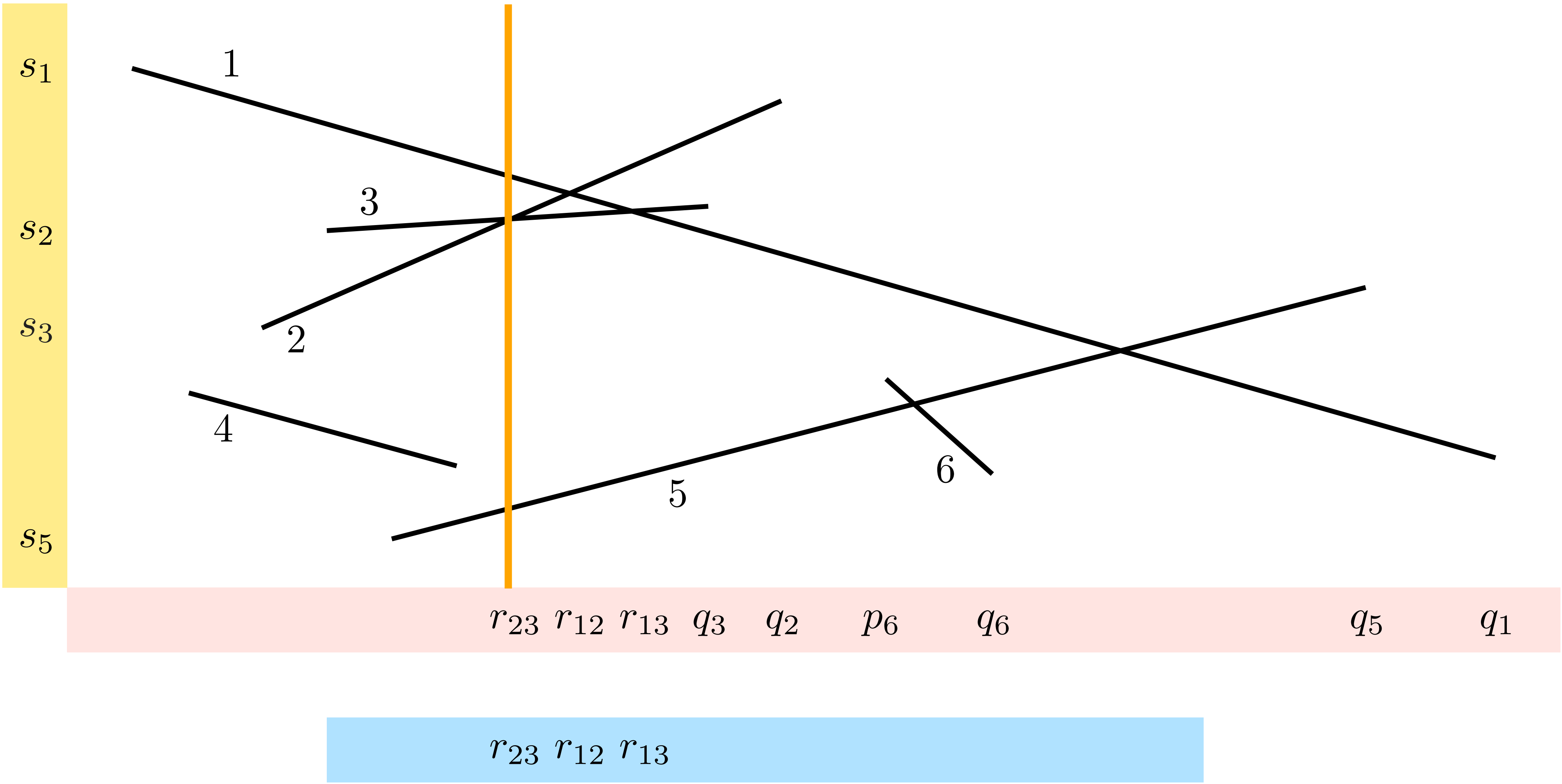
Bentley-Ottman's algorithm: simulation



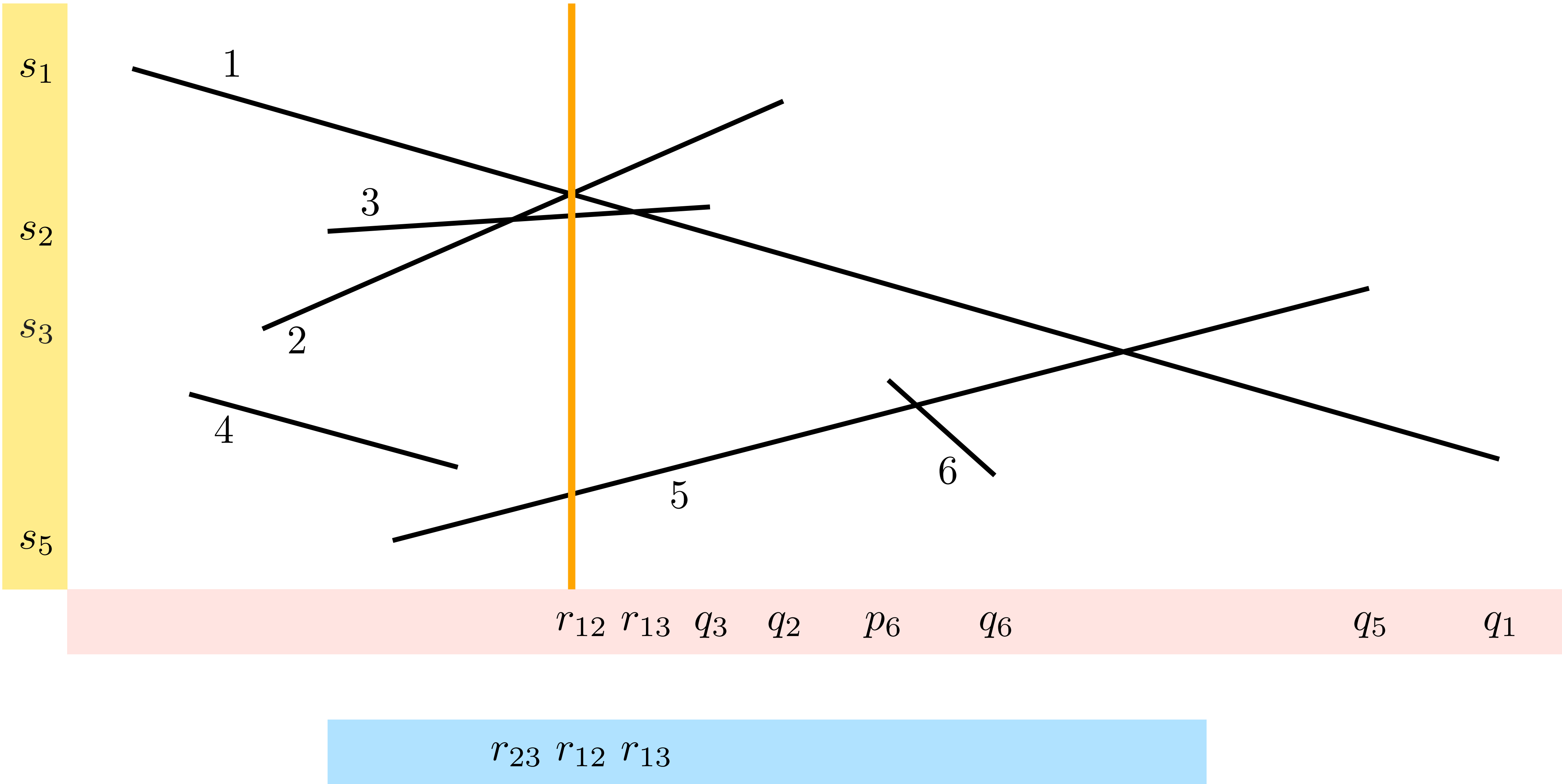
Bentley-Ottman's algorithm: simulation



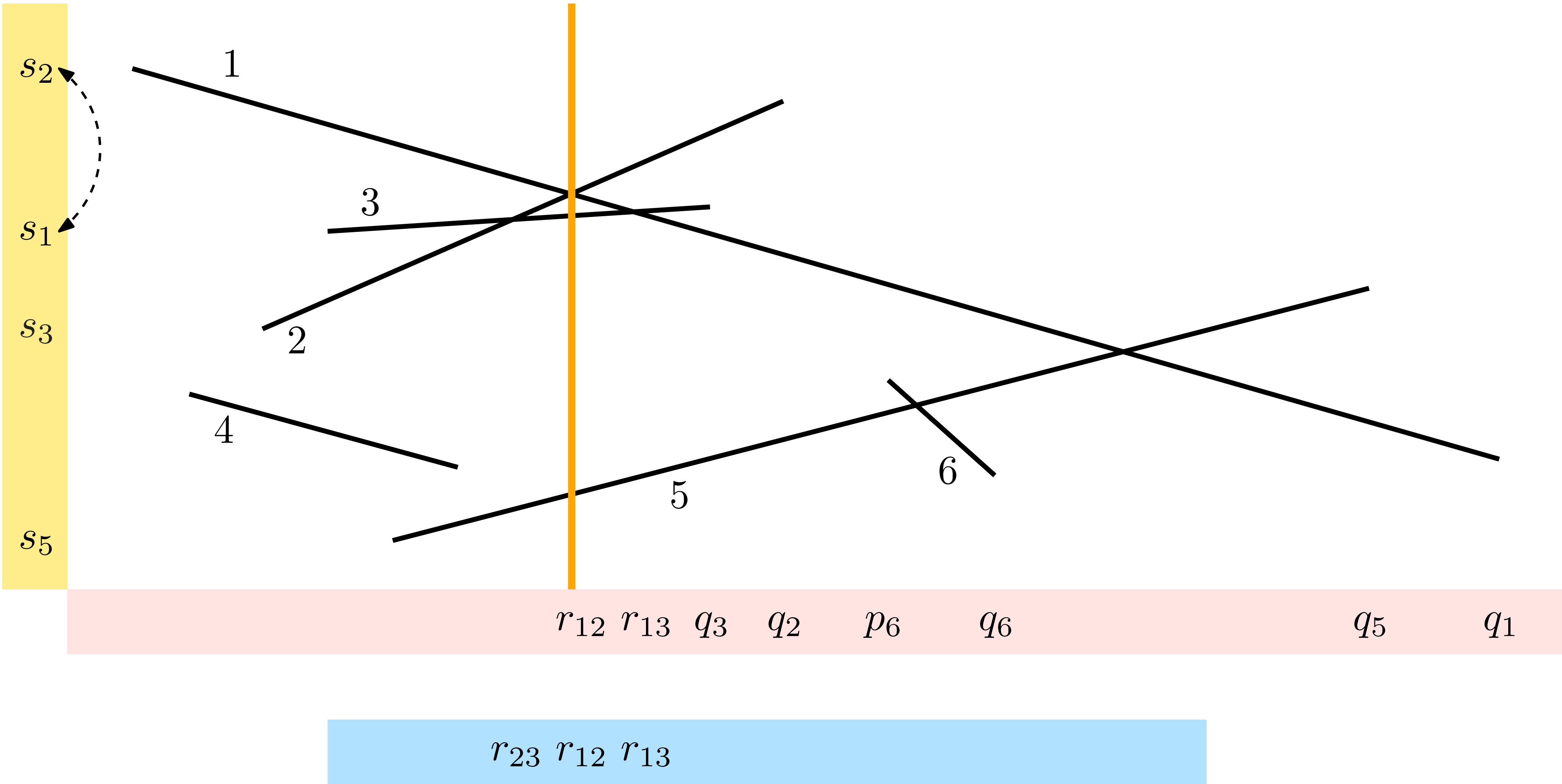
Bentley-Ottman's algorithm: simulation



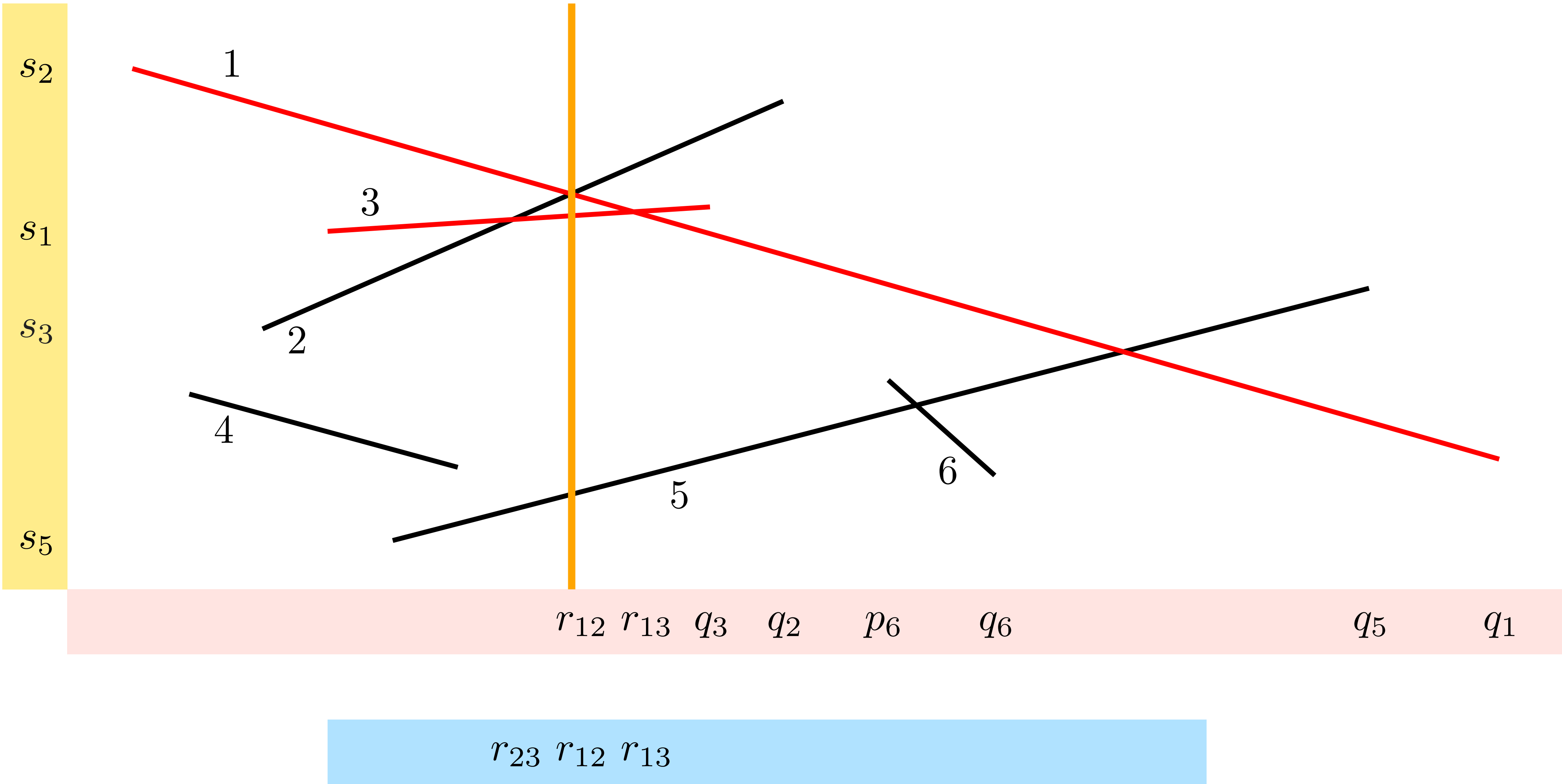
Bentley-Ottman's algorithm: simulation



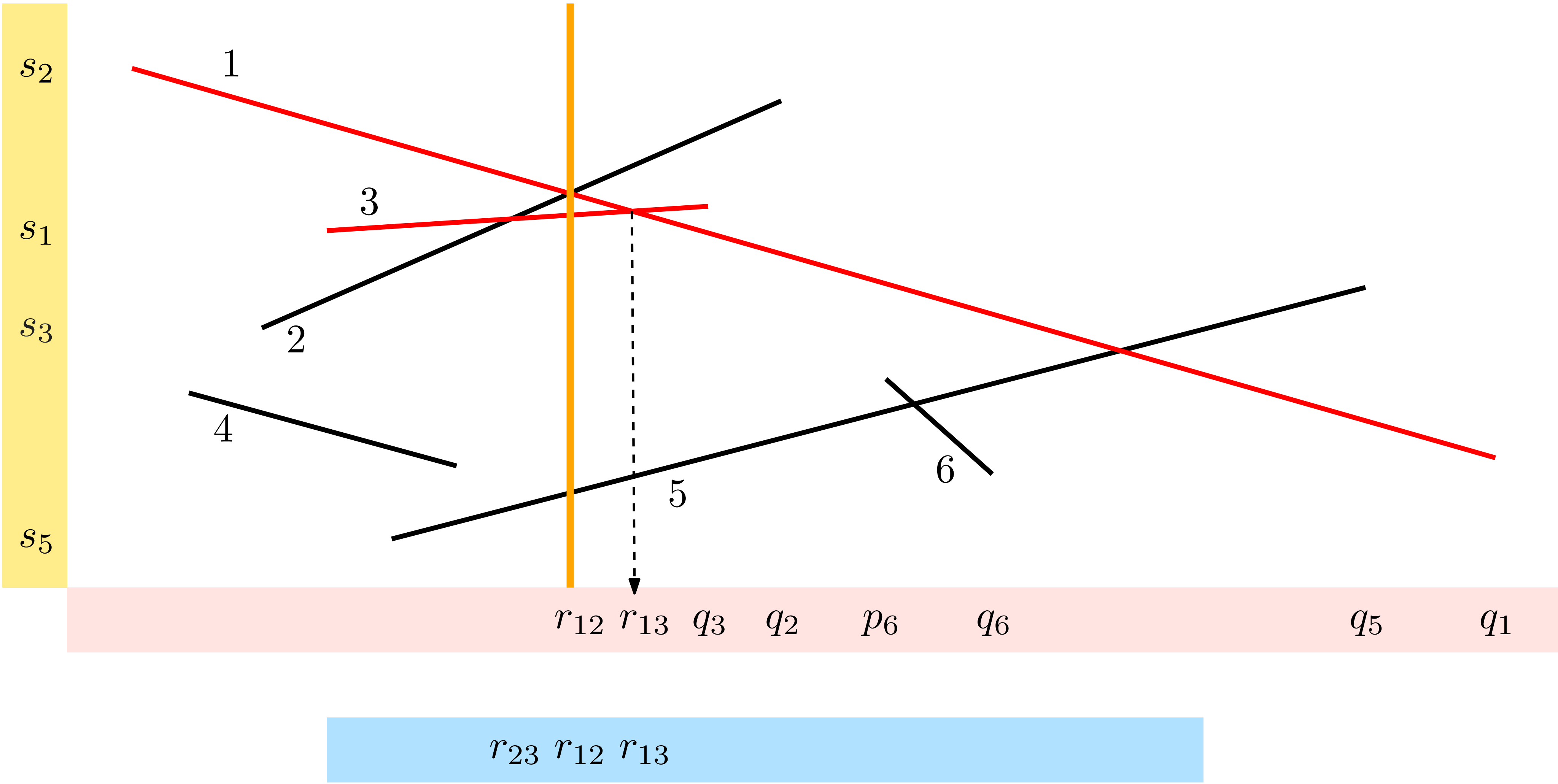
Bentley-Ottman's algorithm: simulation



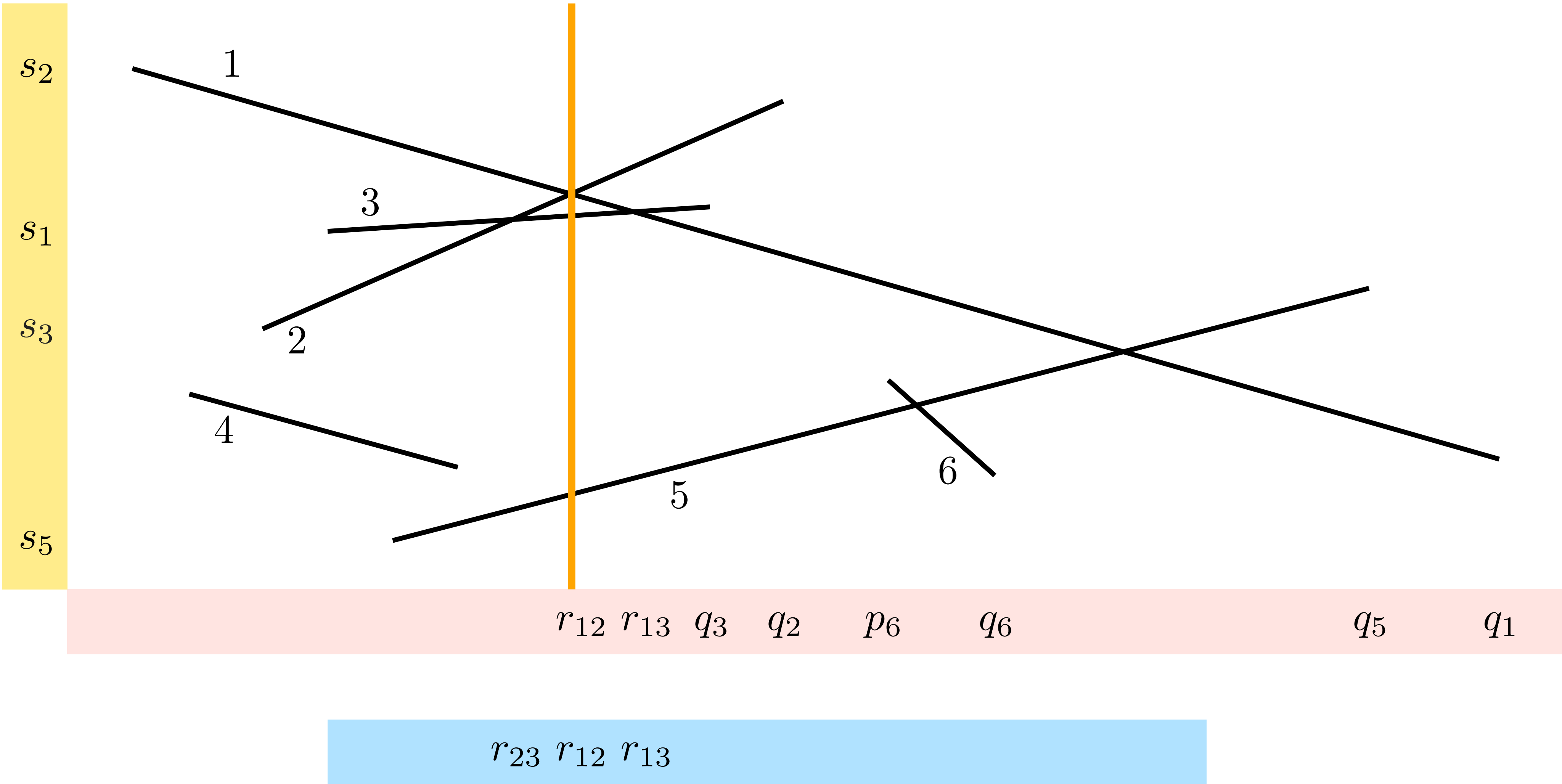
Bentley-Ottman's algorithm: simulation



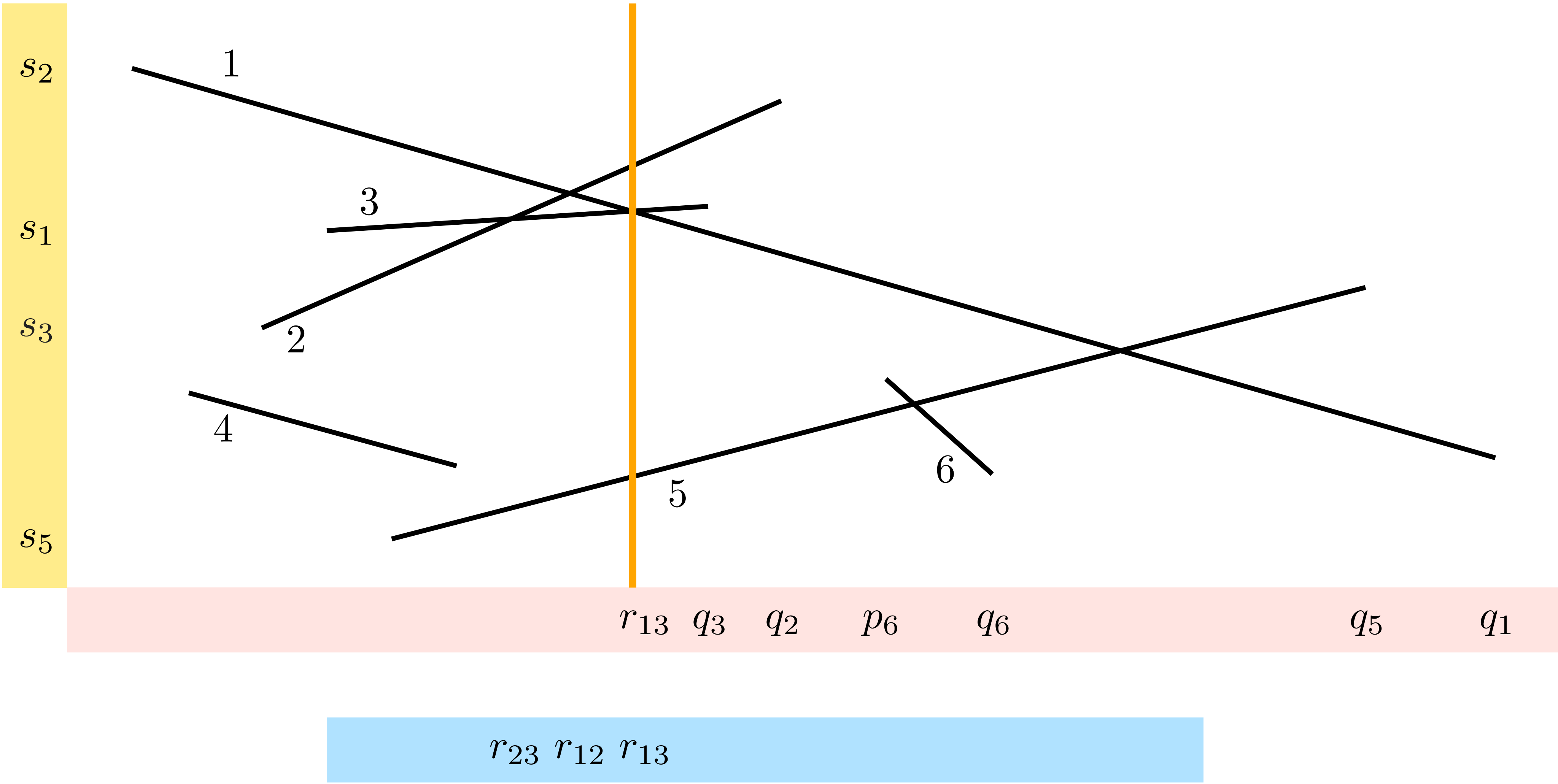
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

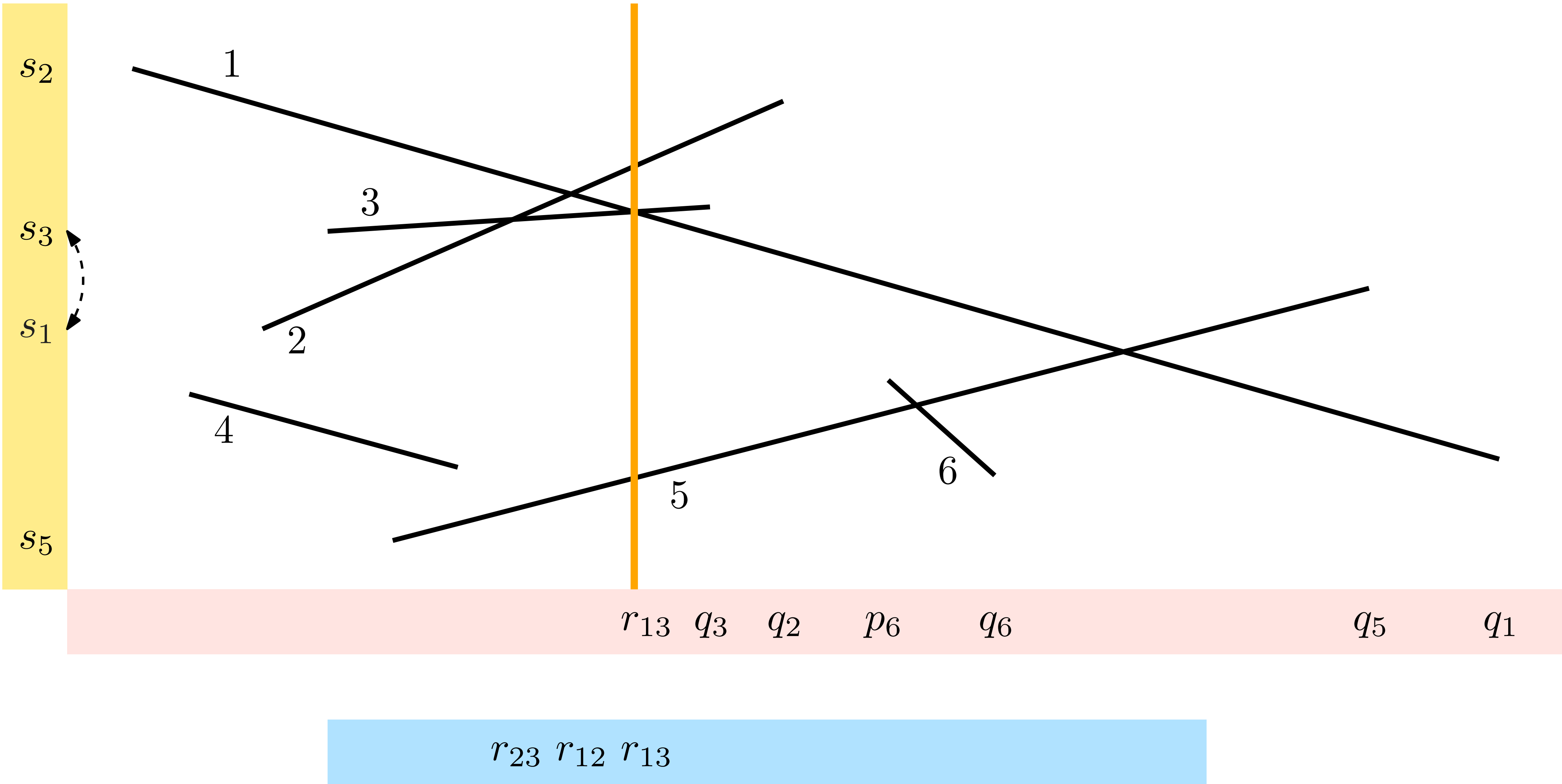


Bentley-Ottman's algorithm: simulation

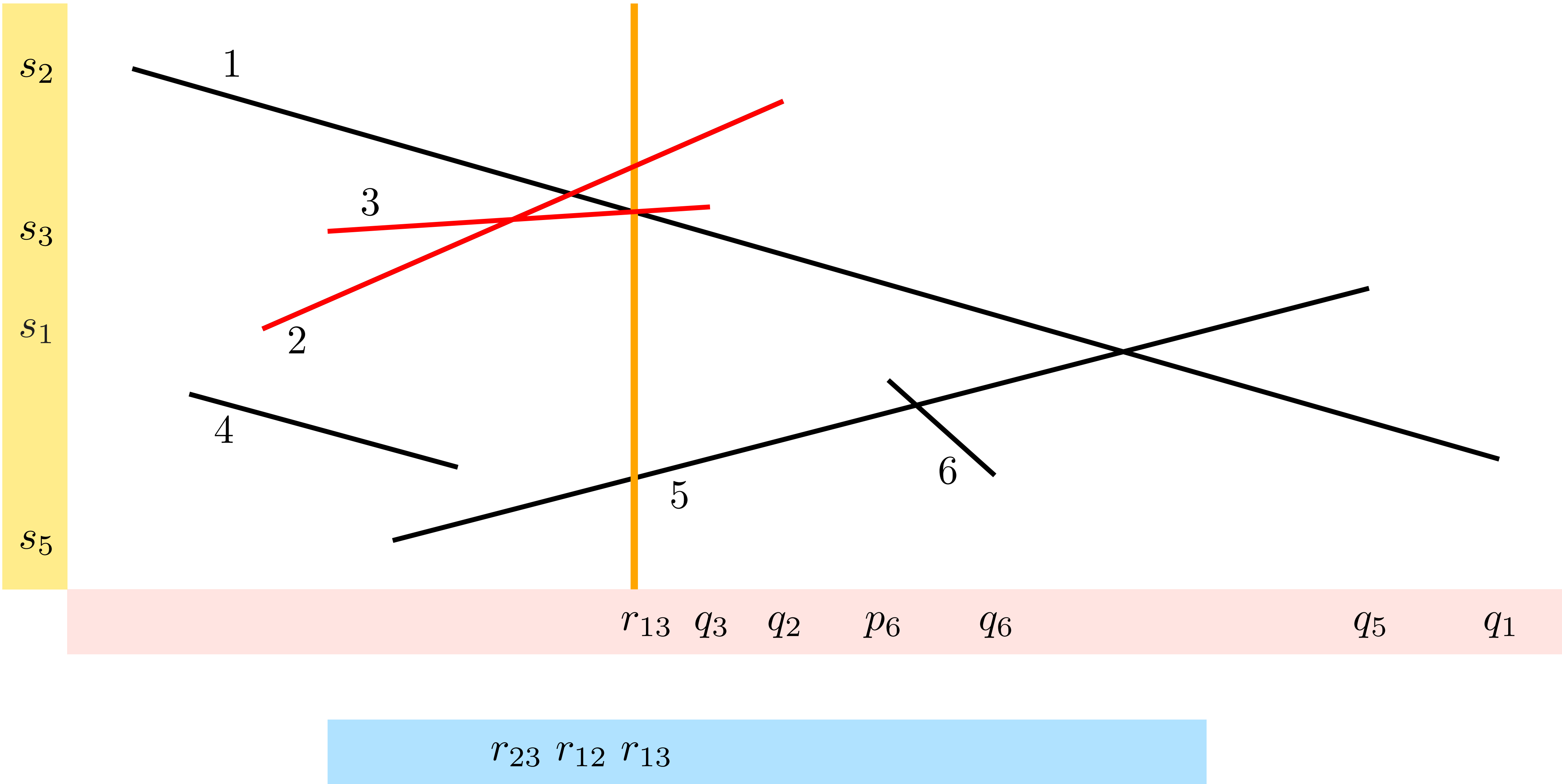


INTERSECTION OF LINE-SEGMENTS

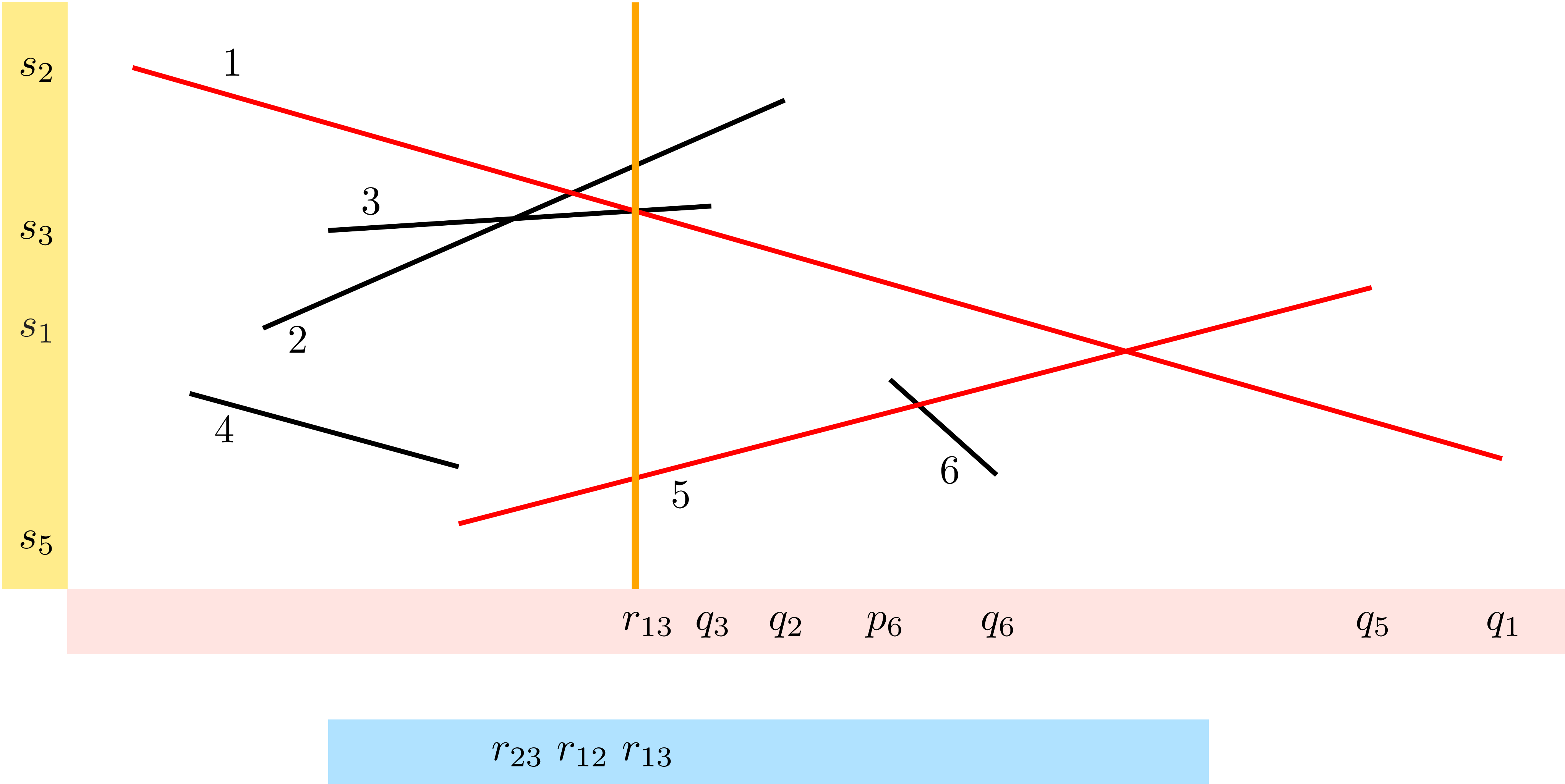
Bentley-Ottman's algorithm: simulation



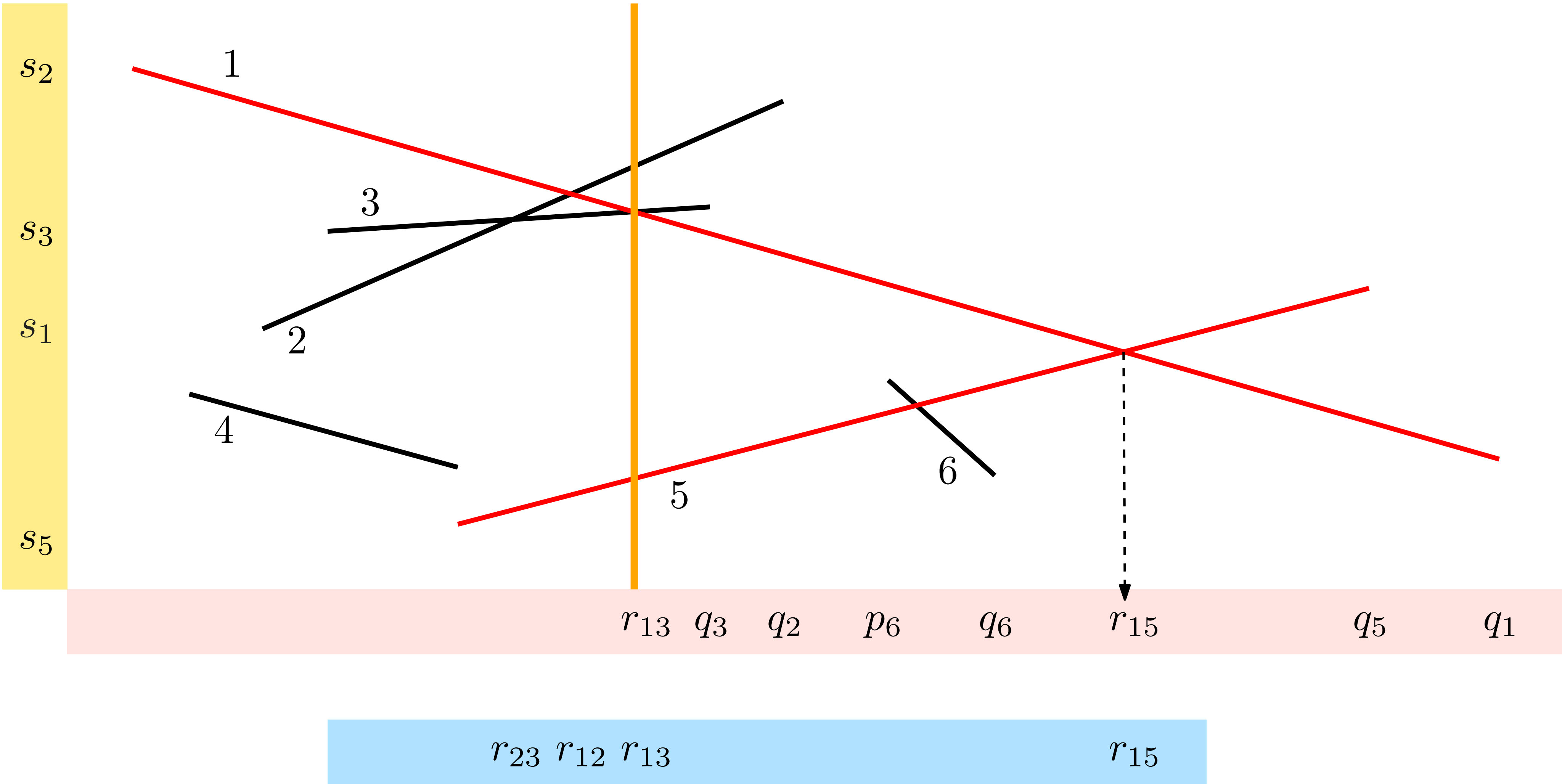
Bentley-Ottman's algorithm: simulation



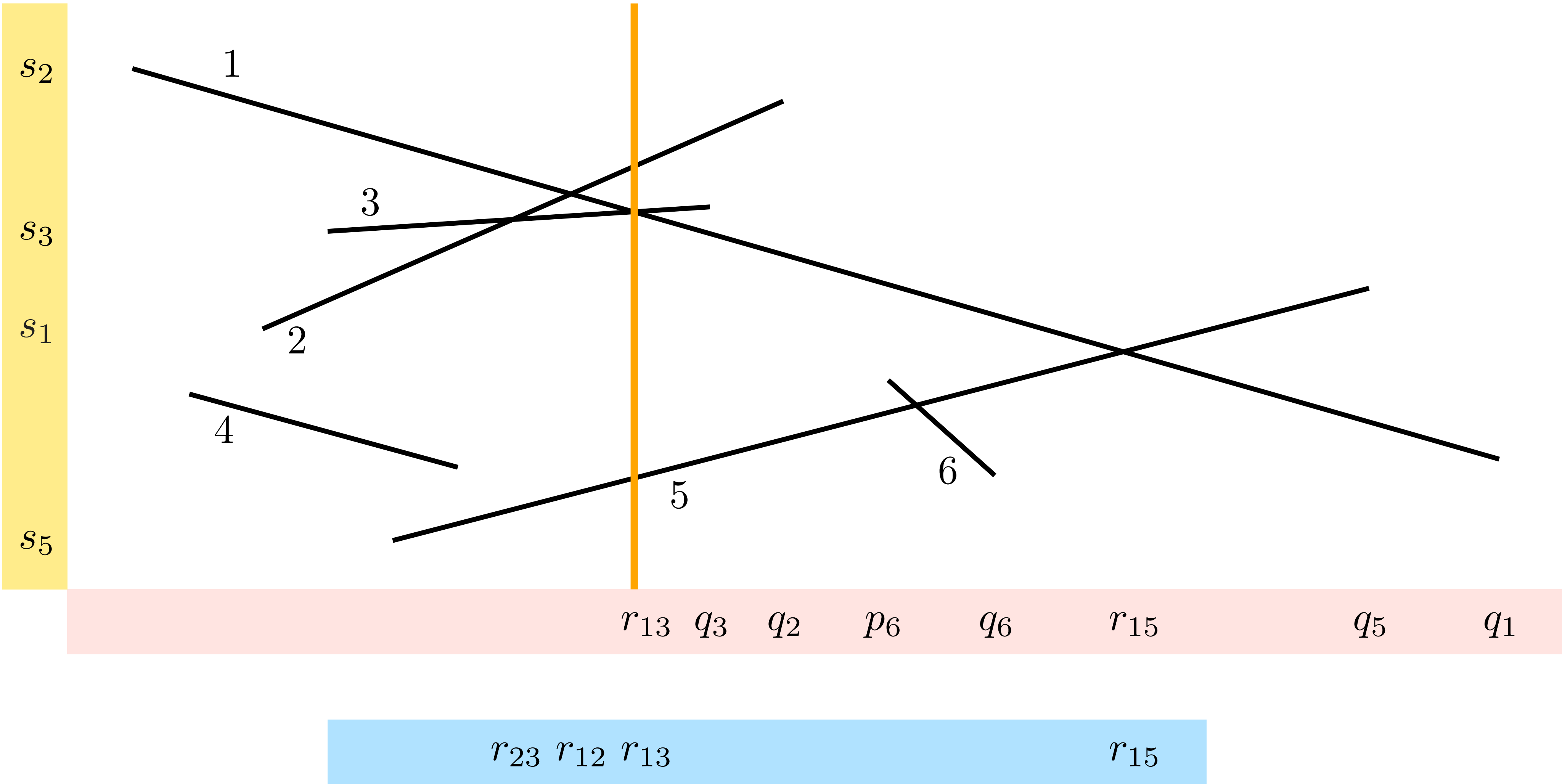
Bentley-Ottman's algorithm: simulation



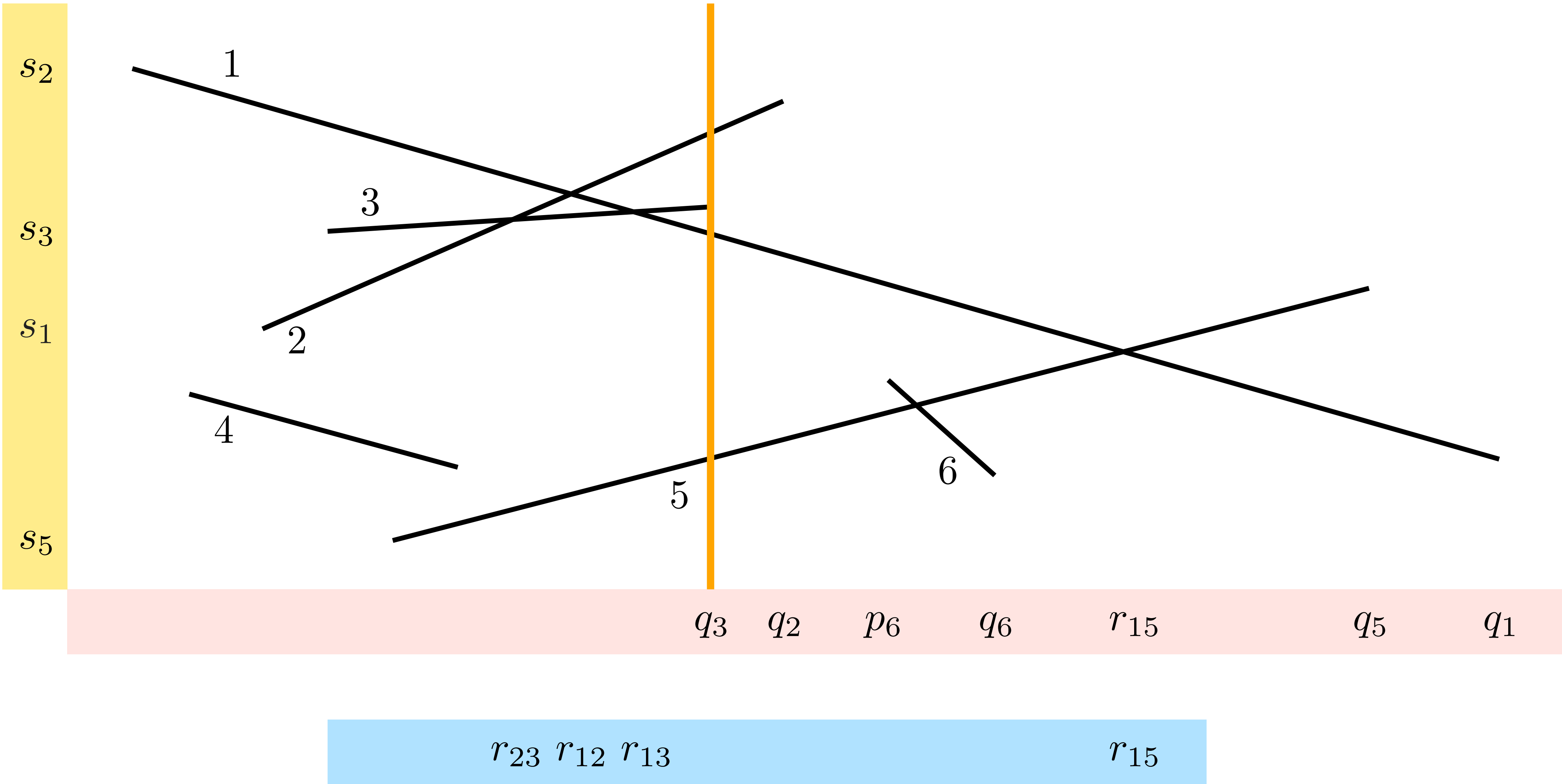
Bentley-Ottman's algorithm: simulation



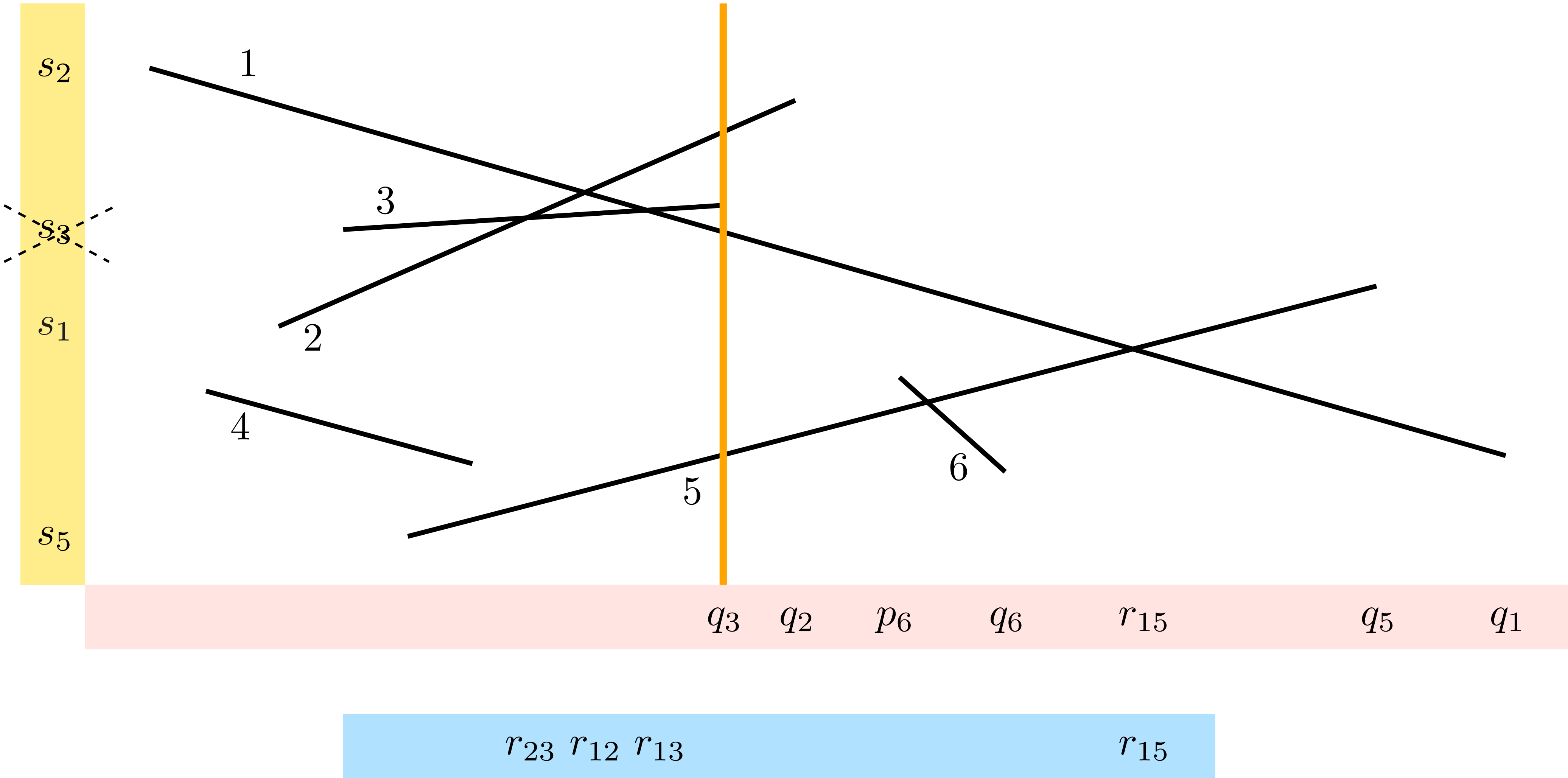
Bentley-Ottman's algorithm: simulation



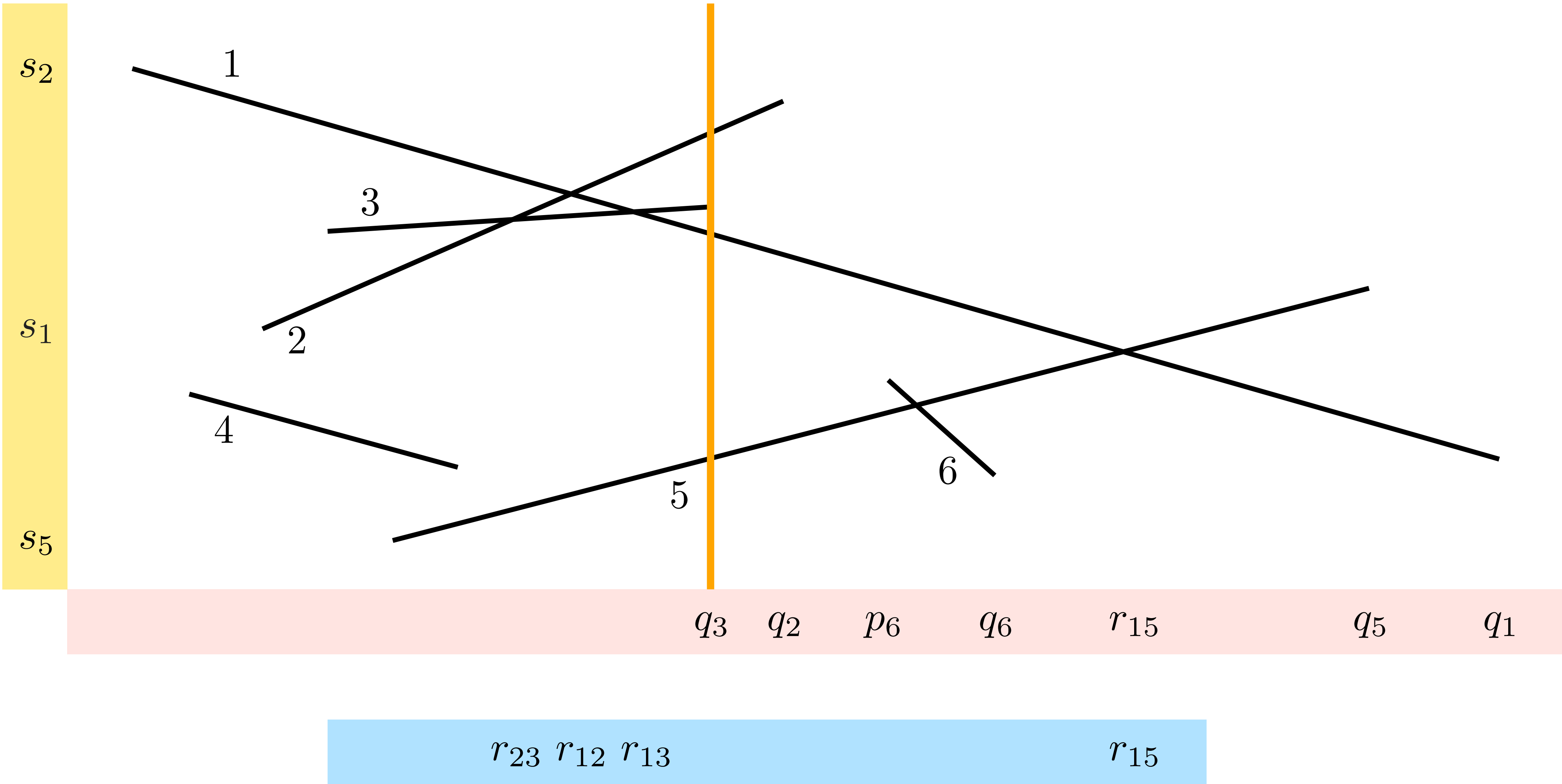
Bentley-Ottman's algorithm: simulation



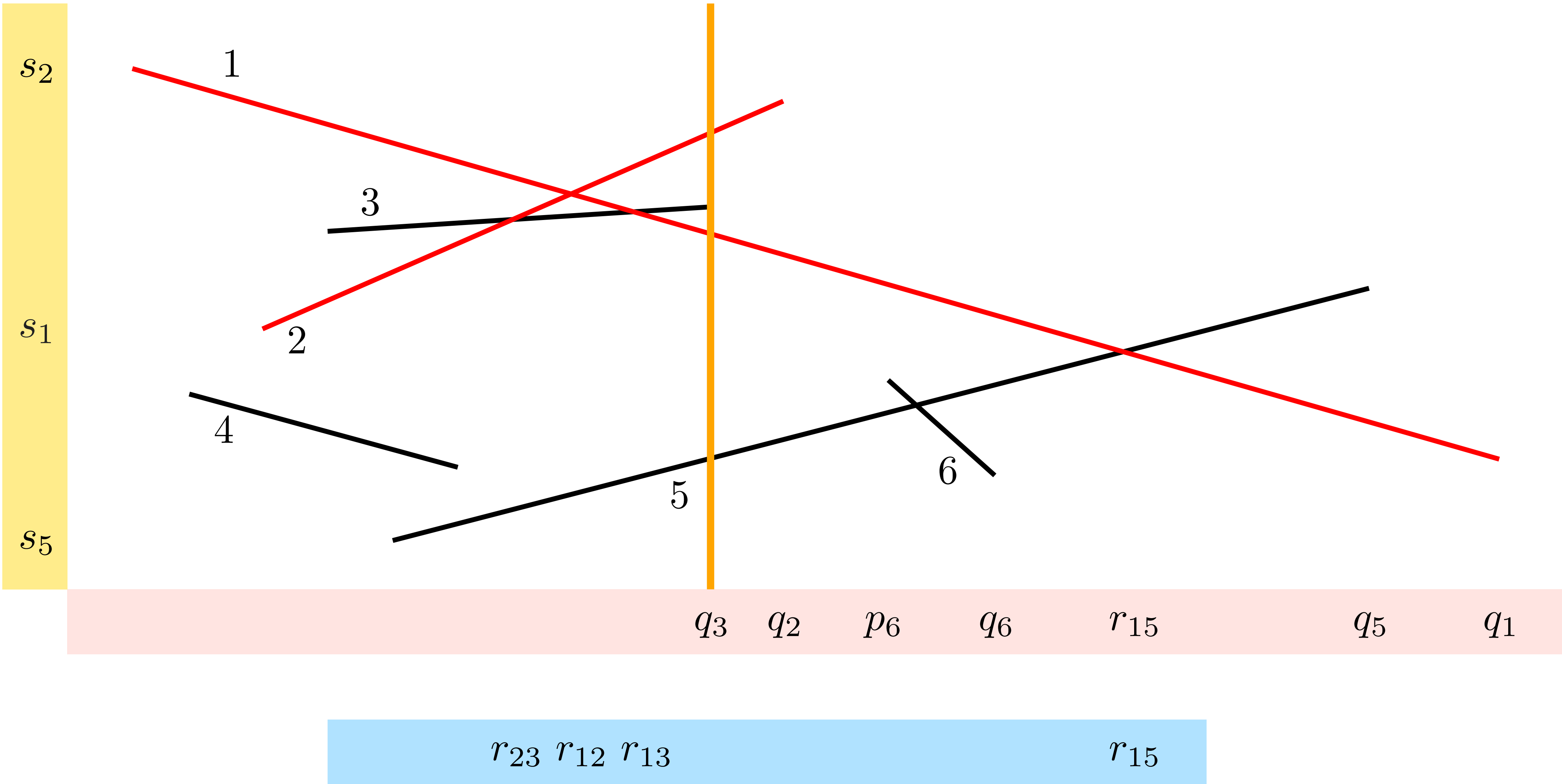
Bentley-Ottman's algorithm: simulation



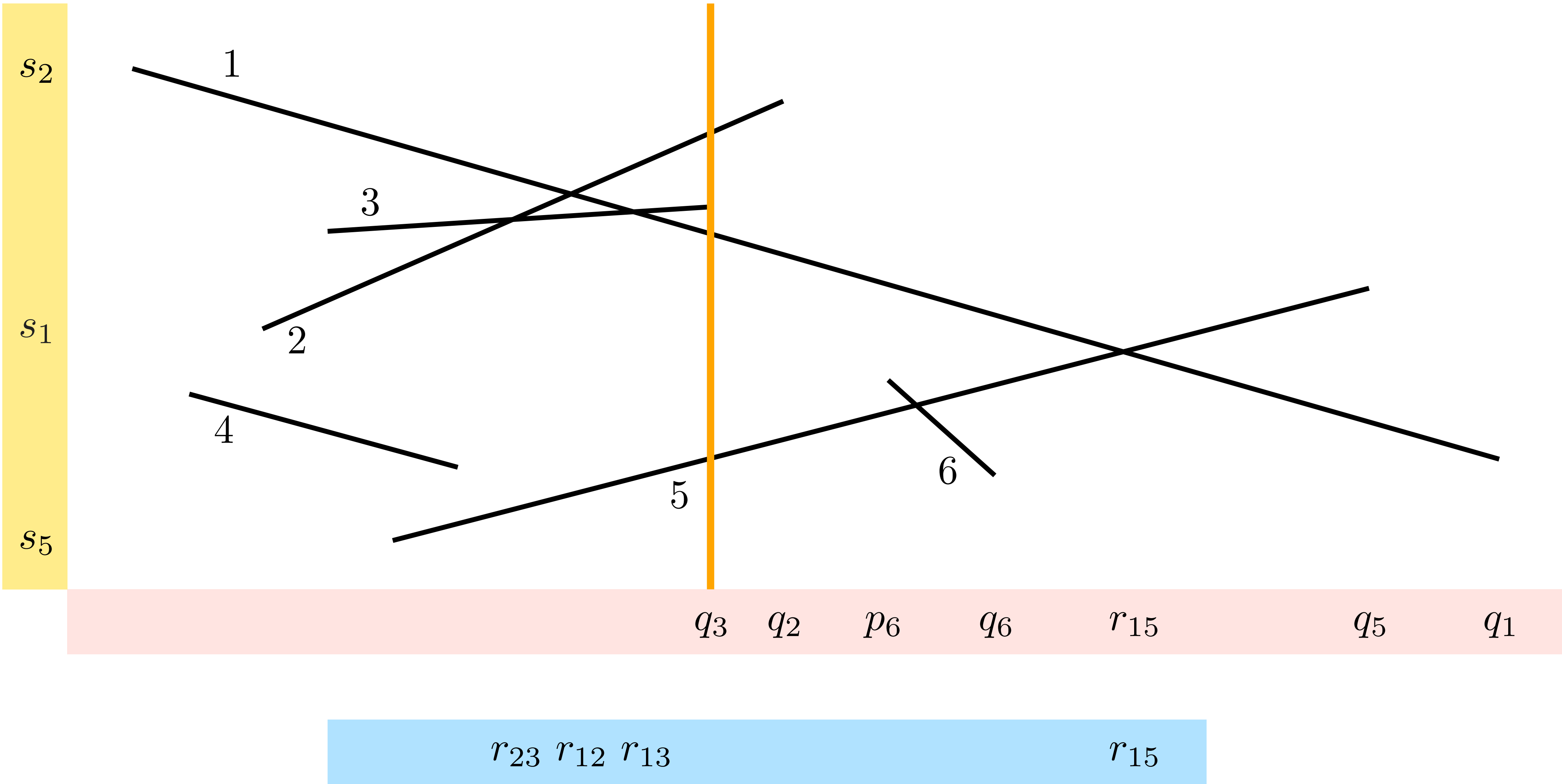
Bentley-Ottman's algorithm: simulation



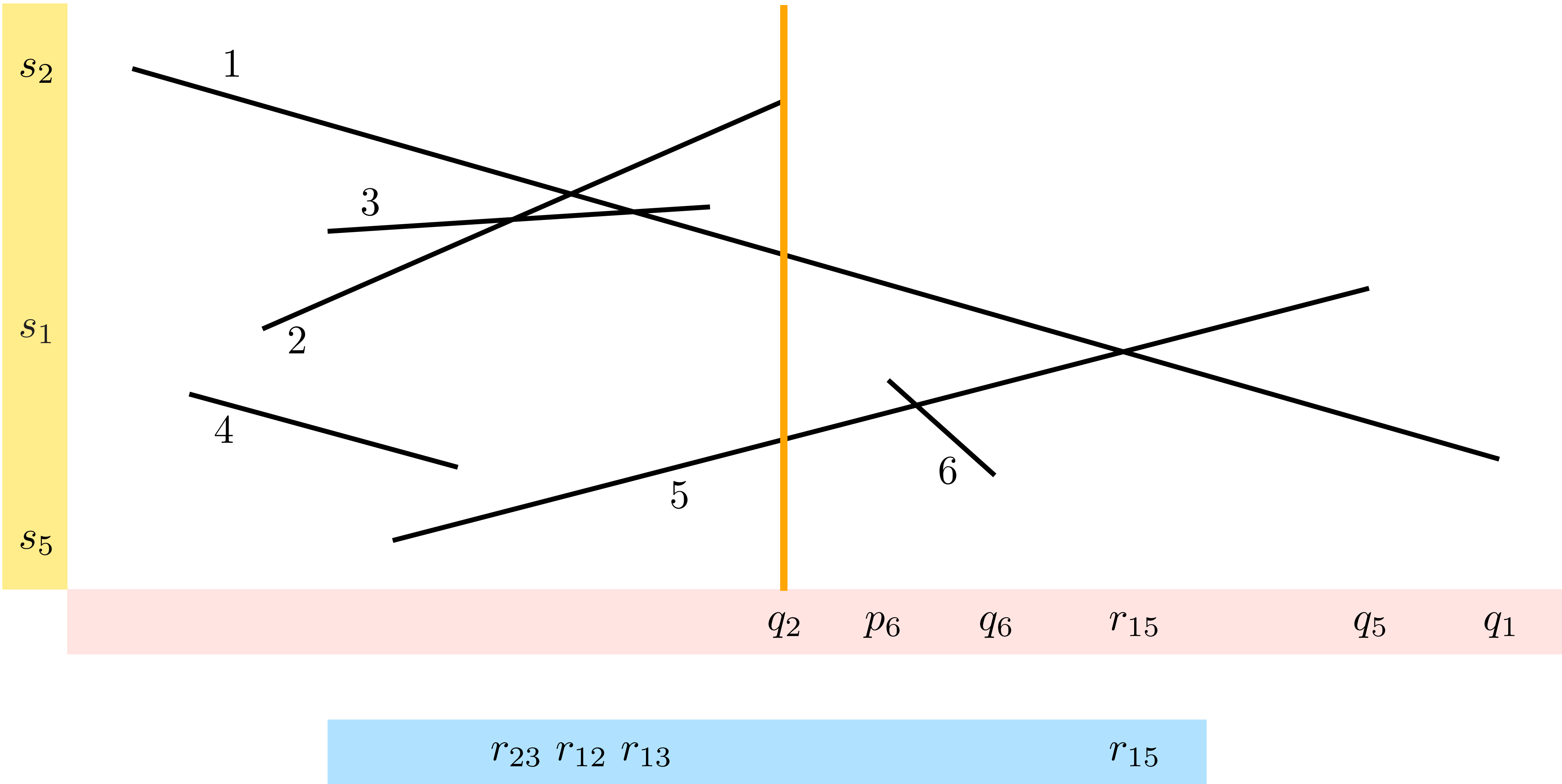
Bentley-Ottman's algorithm: simulation



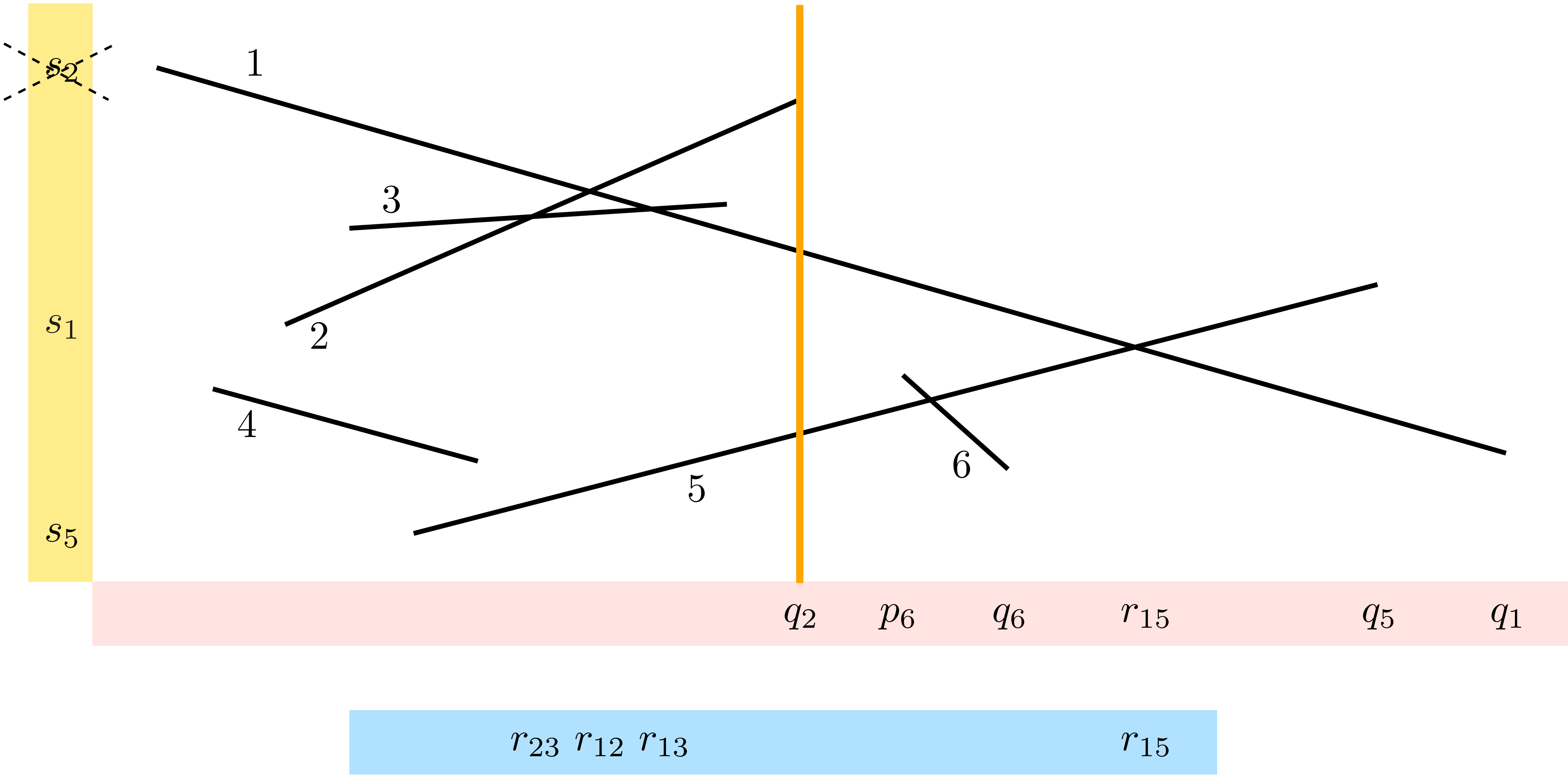
Bentley-Ottman's algorithm: simulation



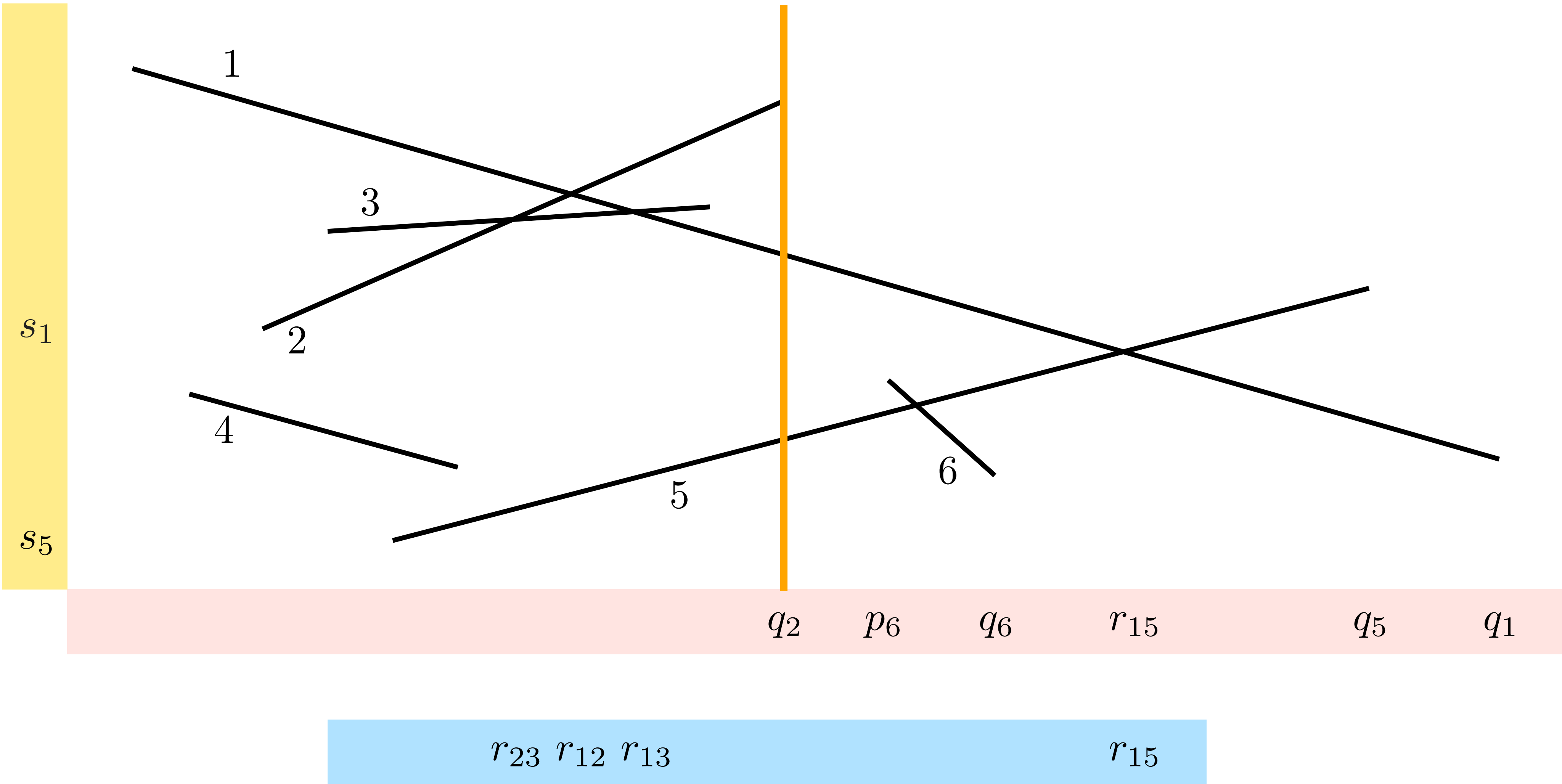
Bentley-Ottman's algorithm: simulation



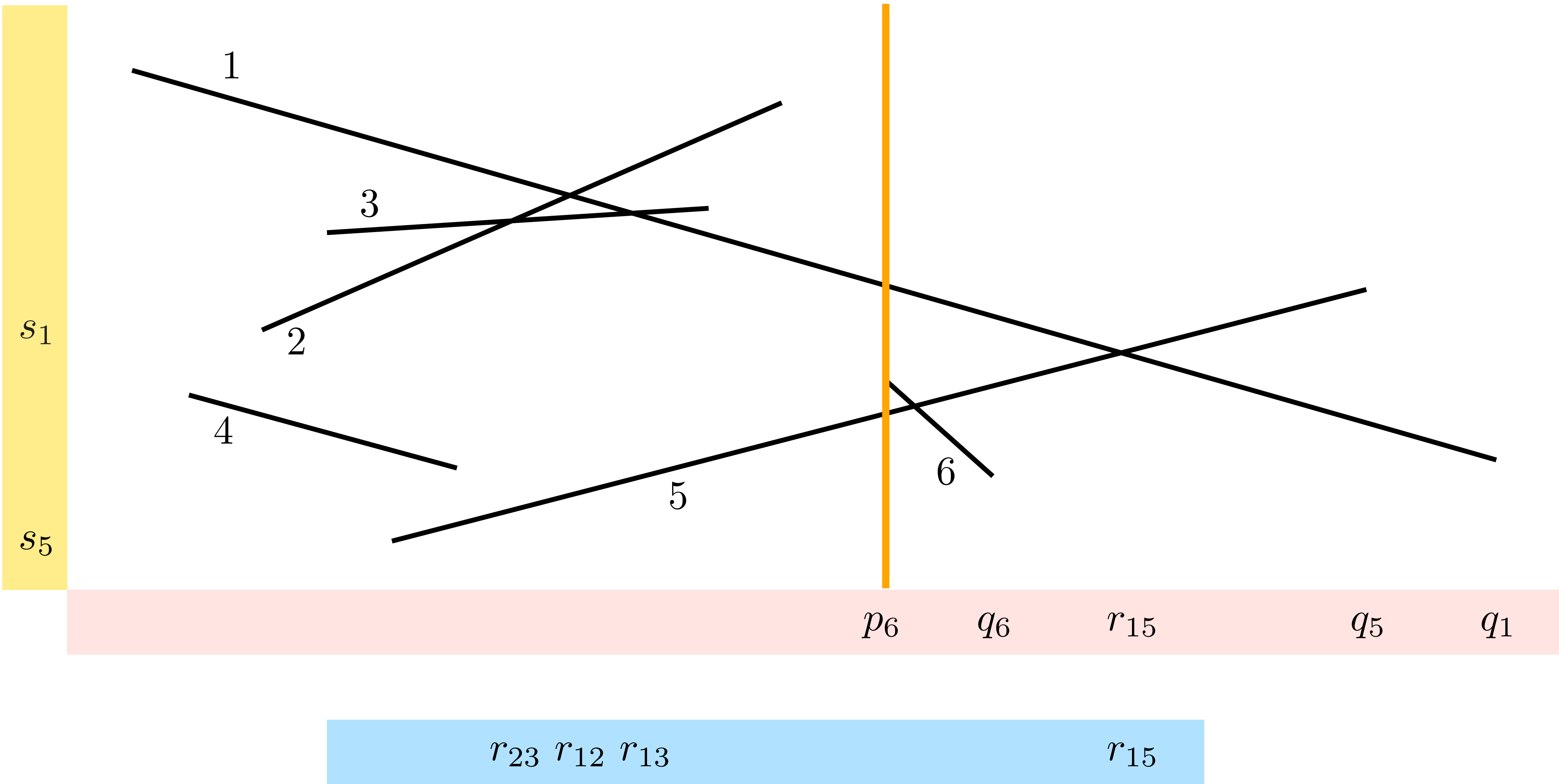
Bentley-Ottman's algorithm: simulation



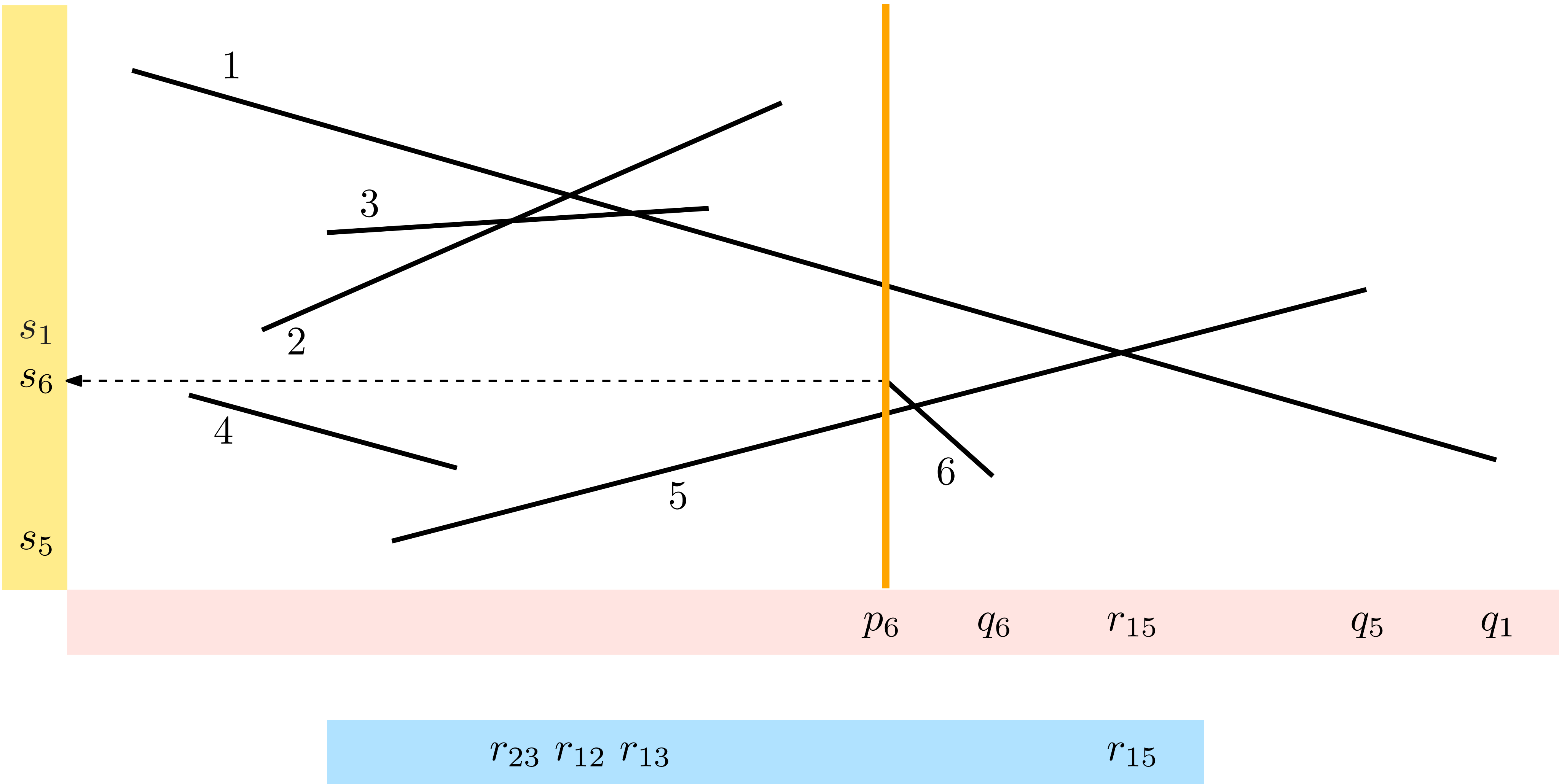
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

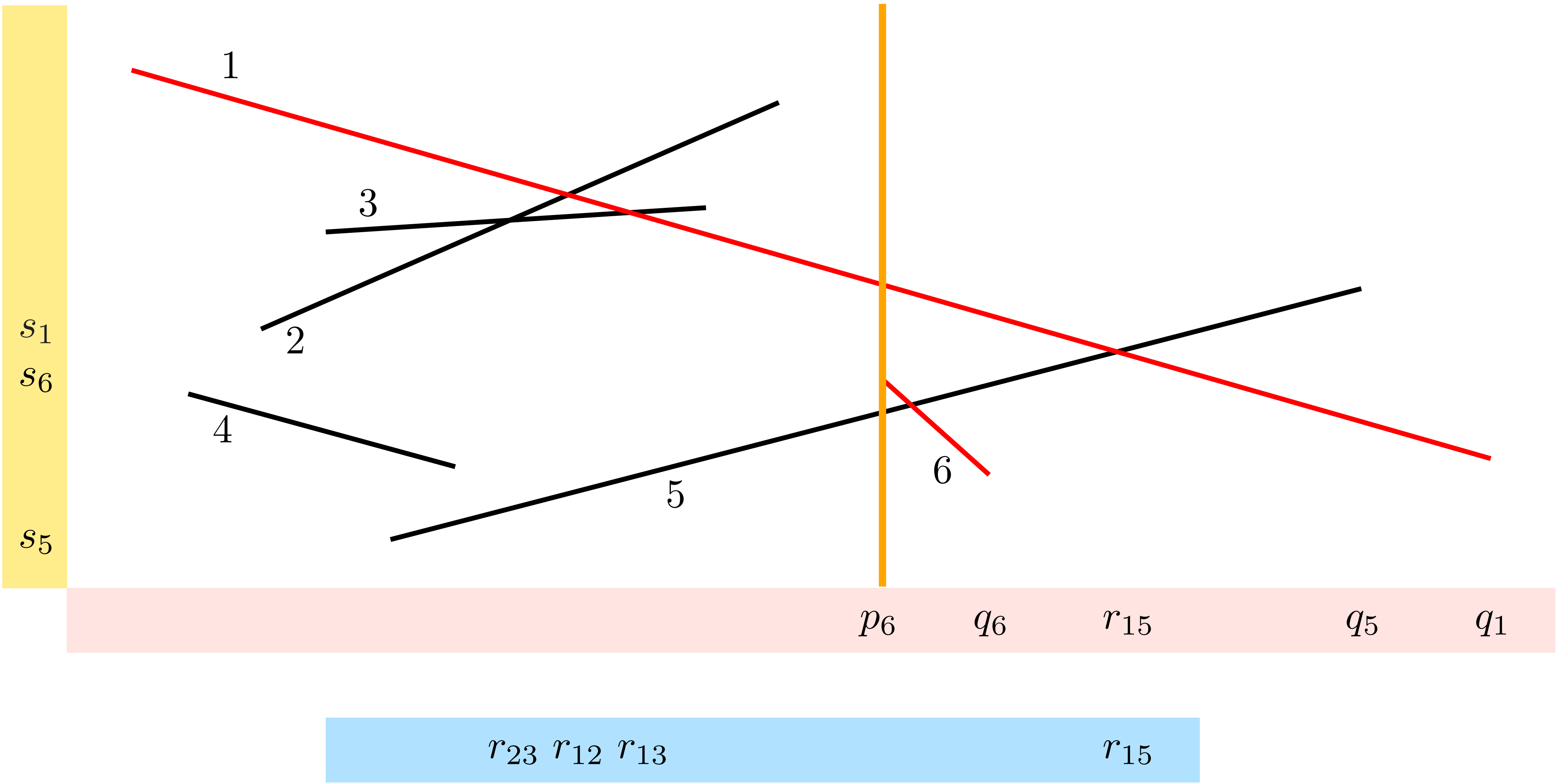


Bentley-Ottman's algorithm: simulation

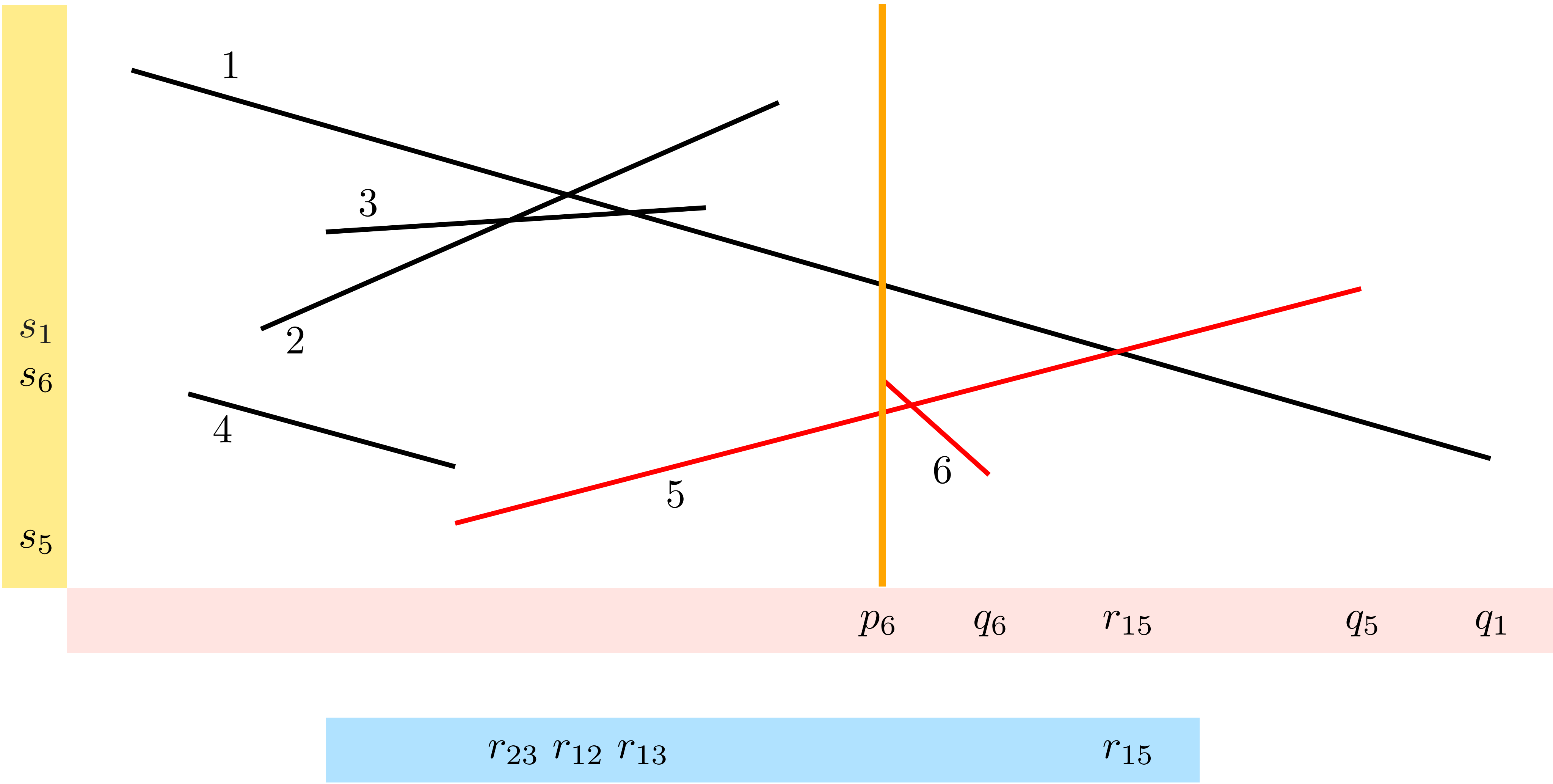


INTERSECTION OF LINE-SEGMENTS

Bentley-Ottman's algorithm: simulation

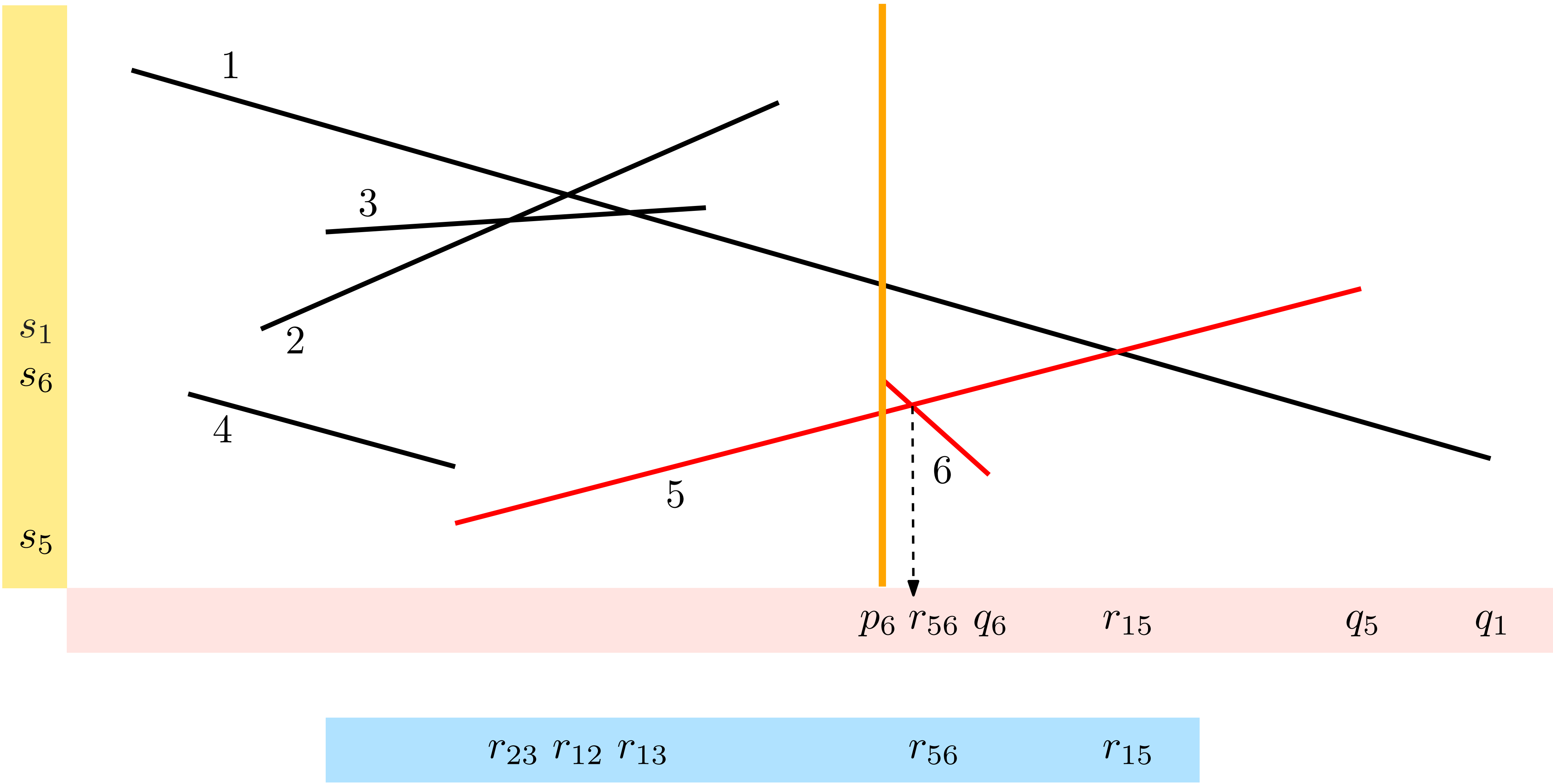


Bentley-Ottman's algorithm: simulation

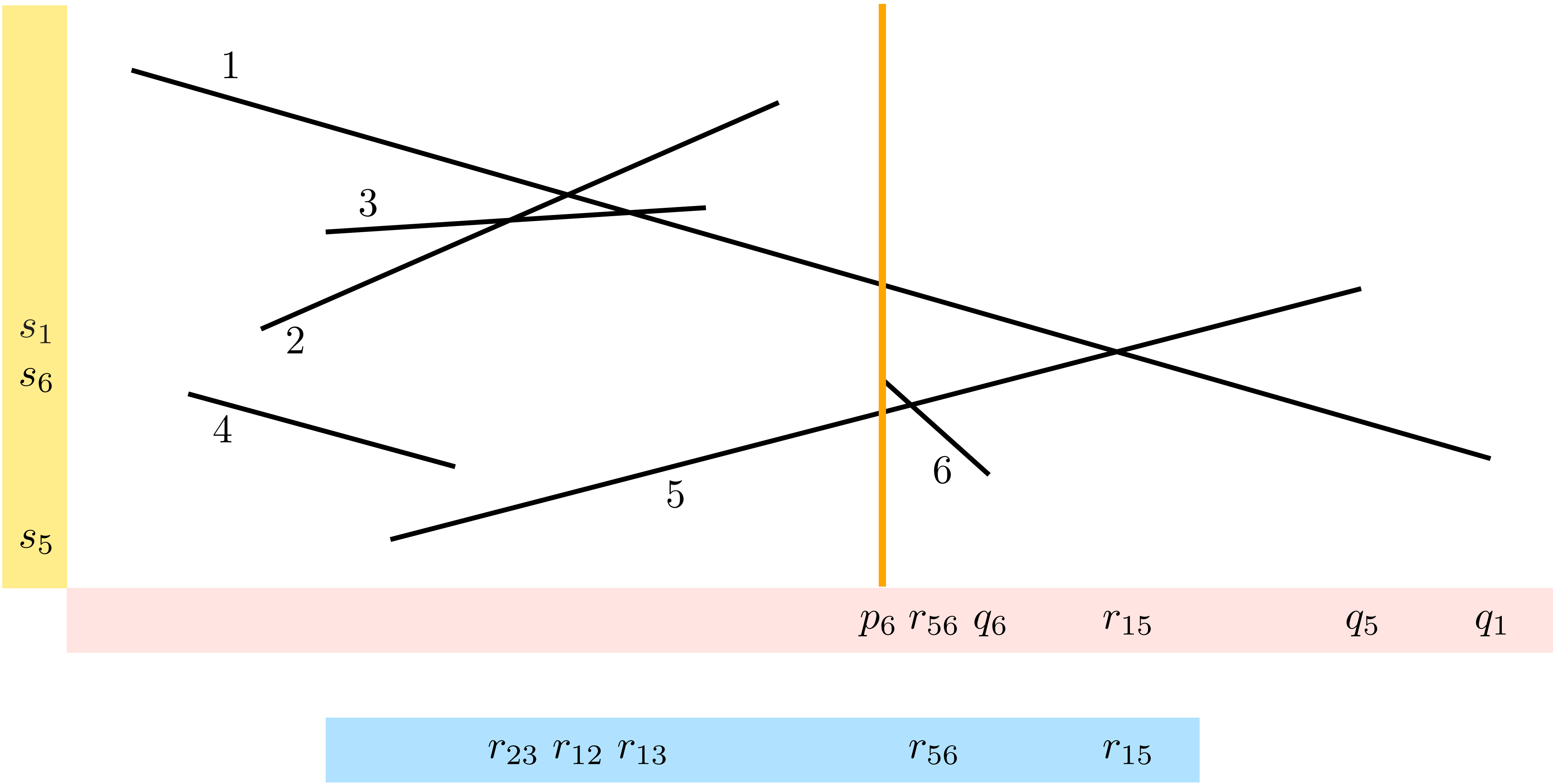


INTERSECTION OF LINE-SEGMENTS

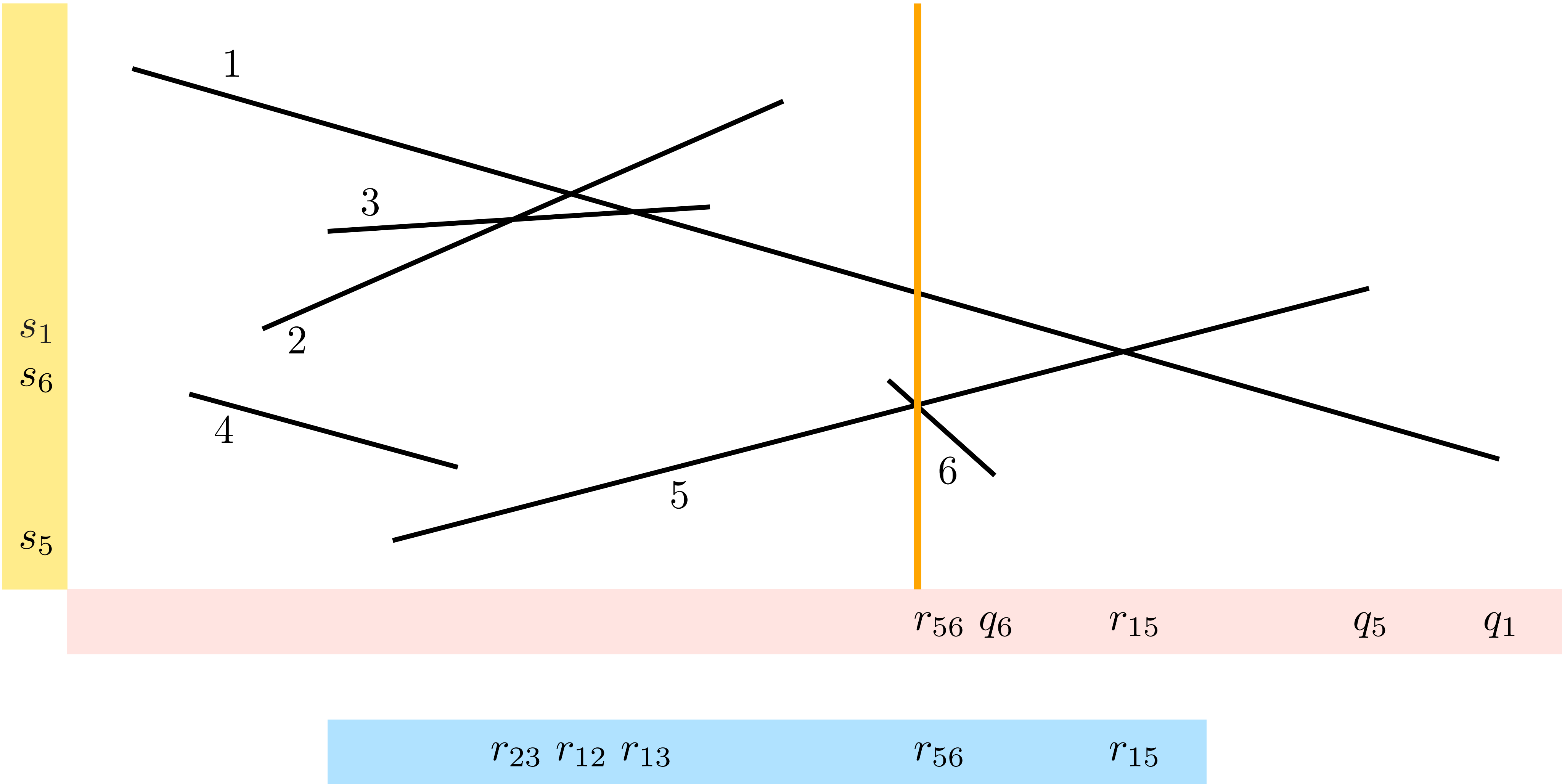
Bentley-Ottman's algorithm: simulation



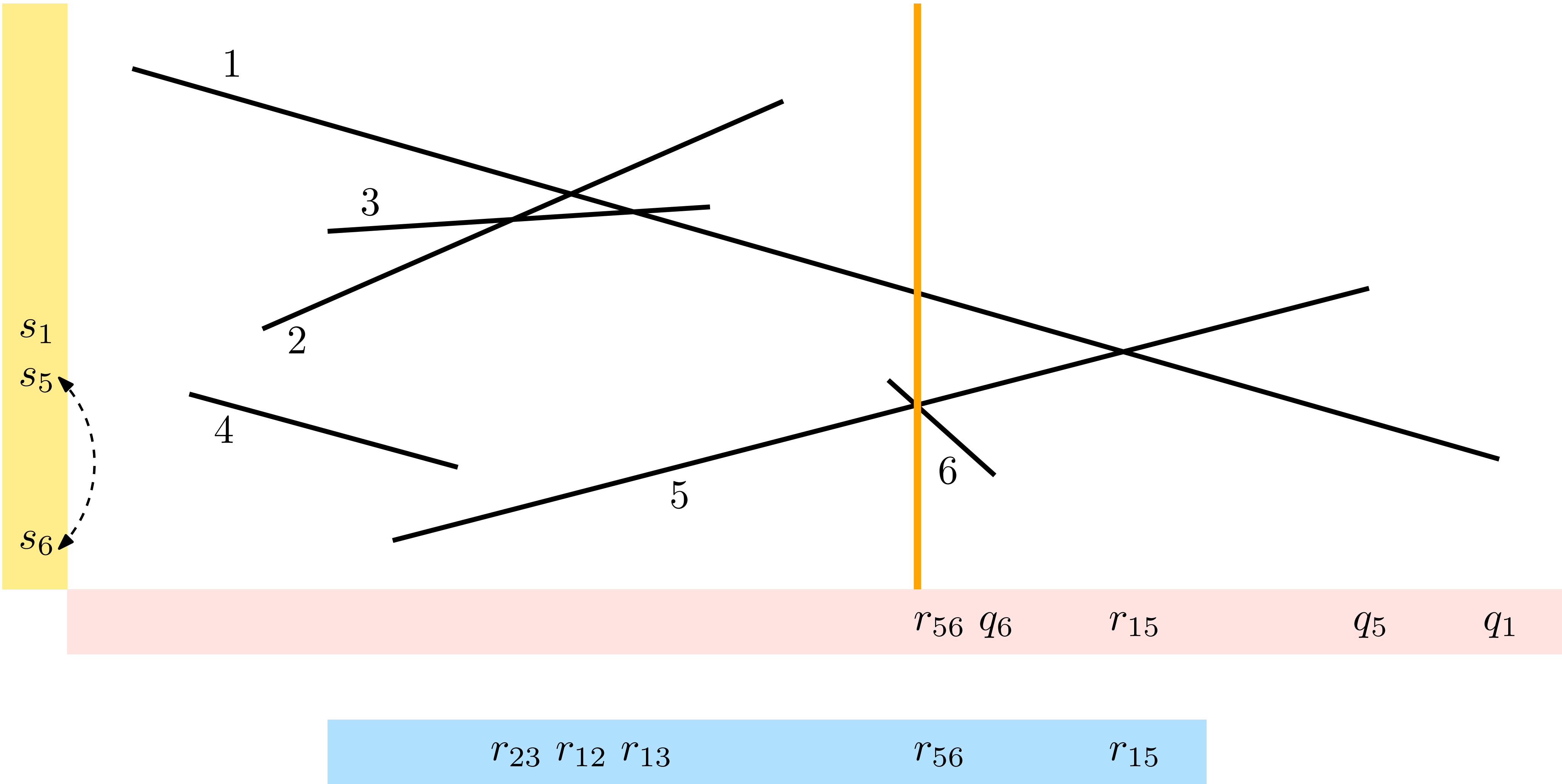
Bentley-Ottman's algorithm: simulation



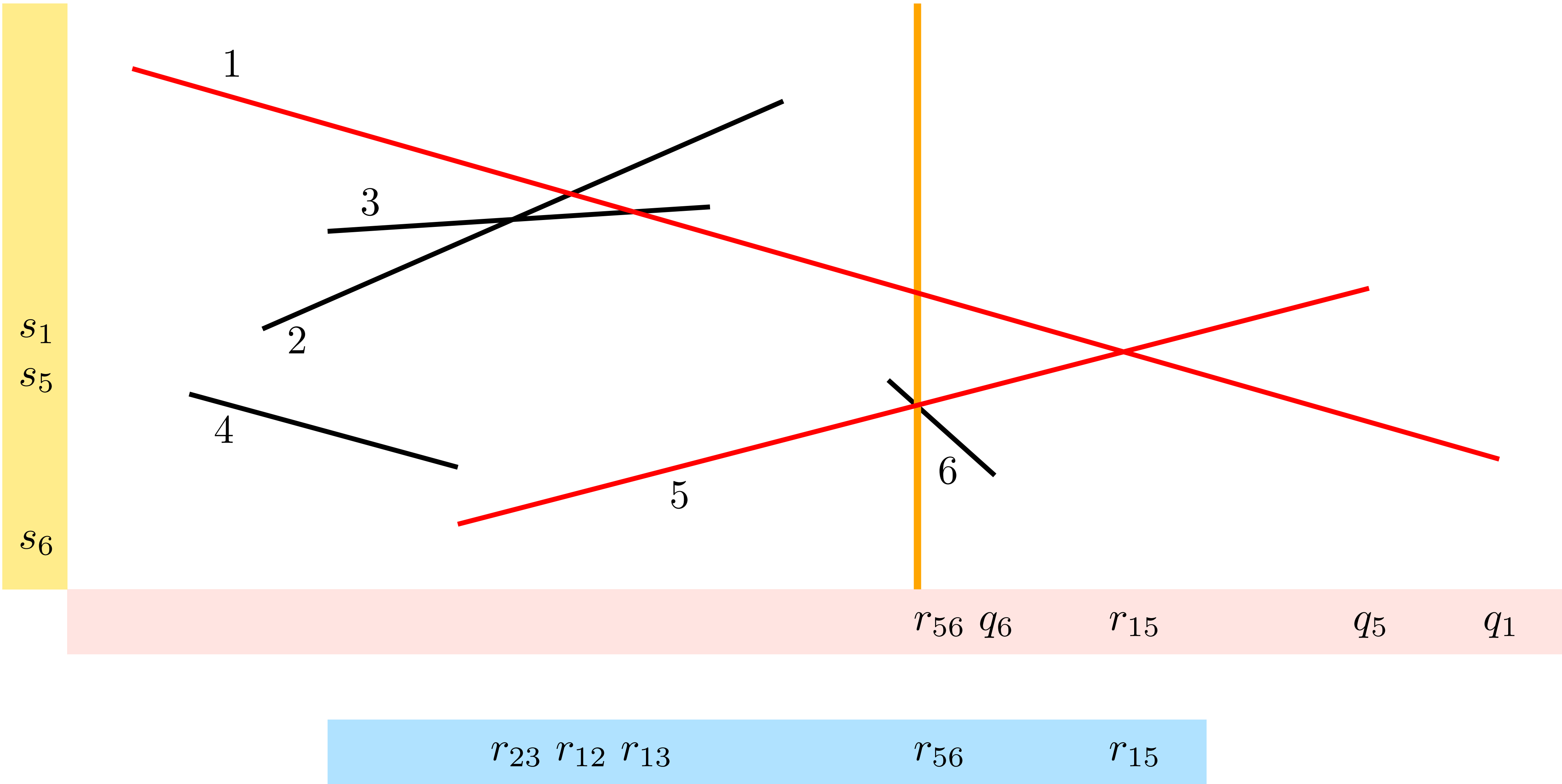
Bentley-Ottman's algorithm: simulation



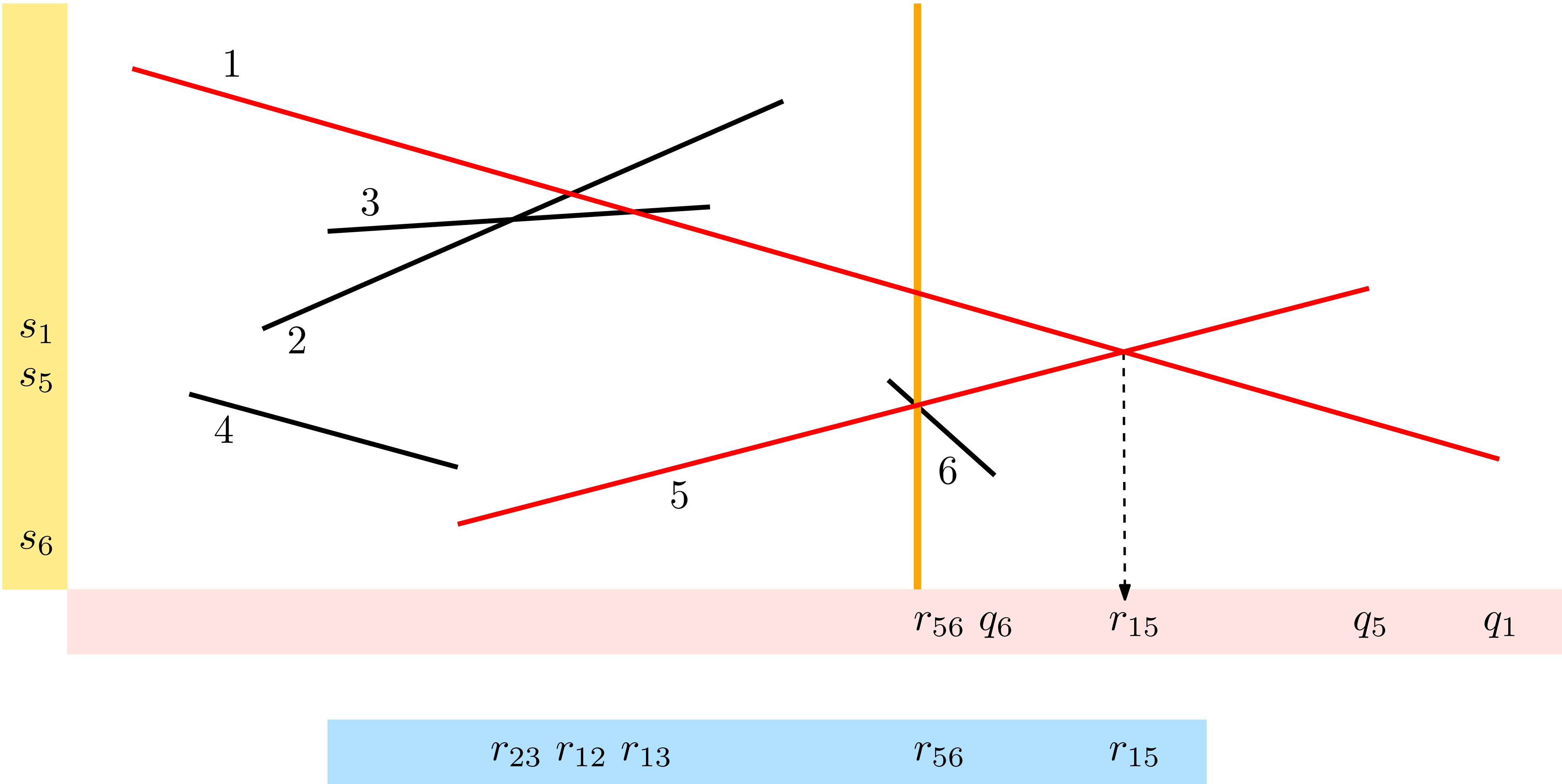
Bentley-Ottman's algorithm: simulation



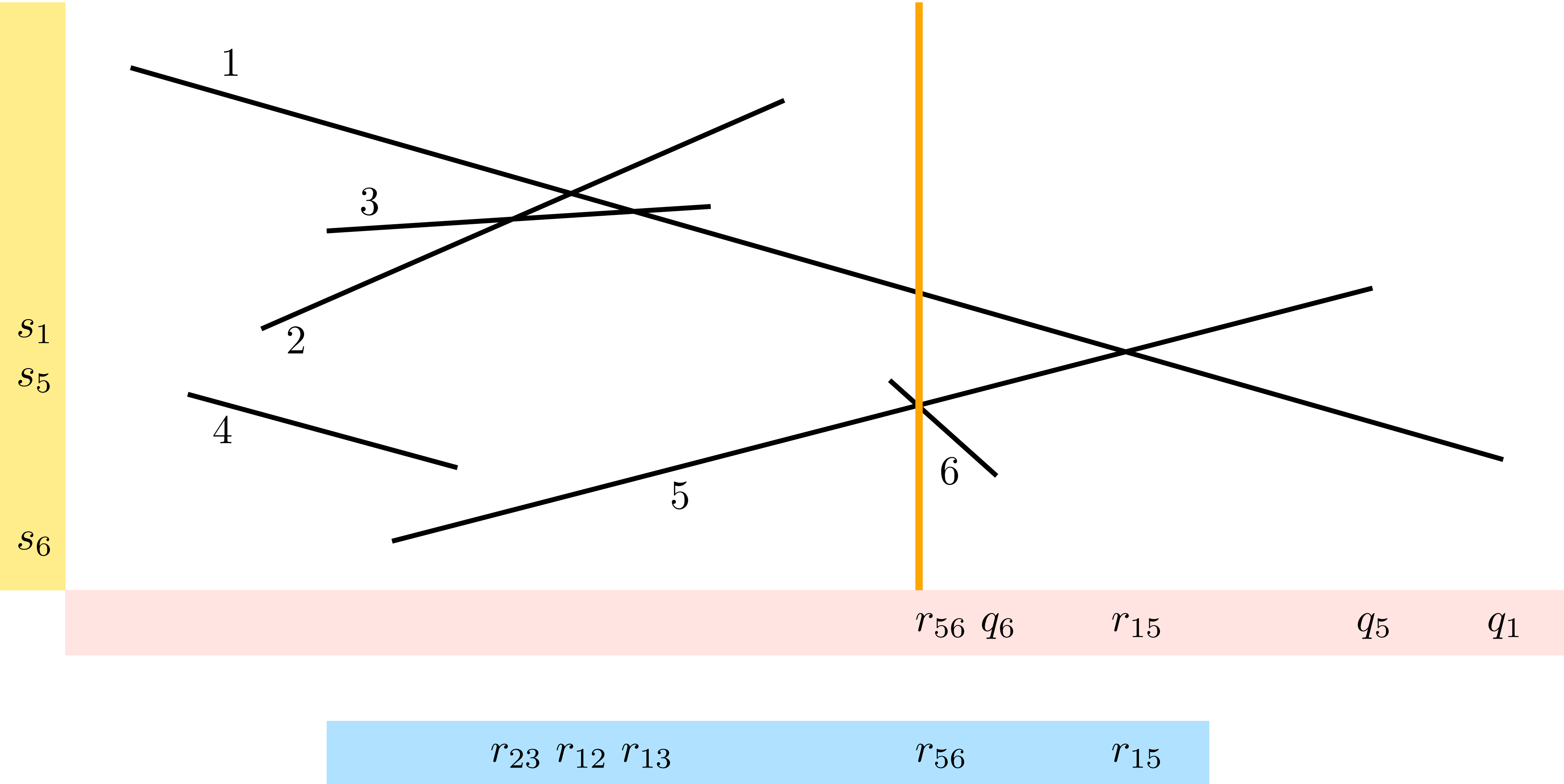
Bentley-Ottman's algorithm: simulation



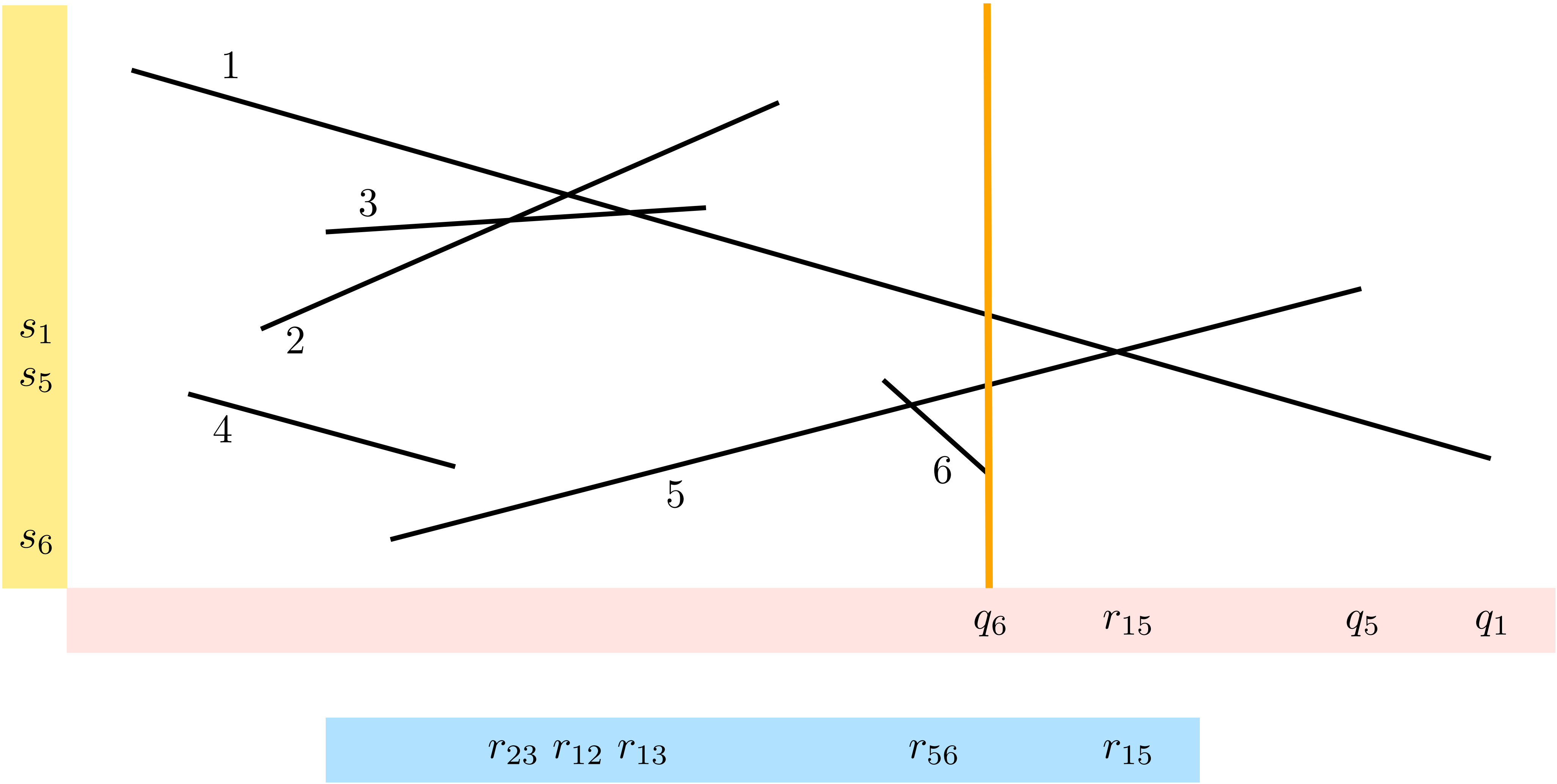
Bentley-Ottman's algorithm: simulation



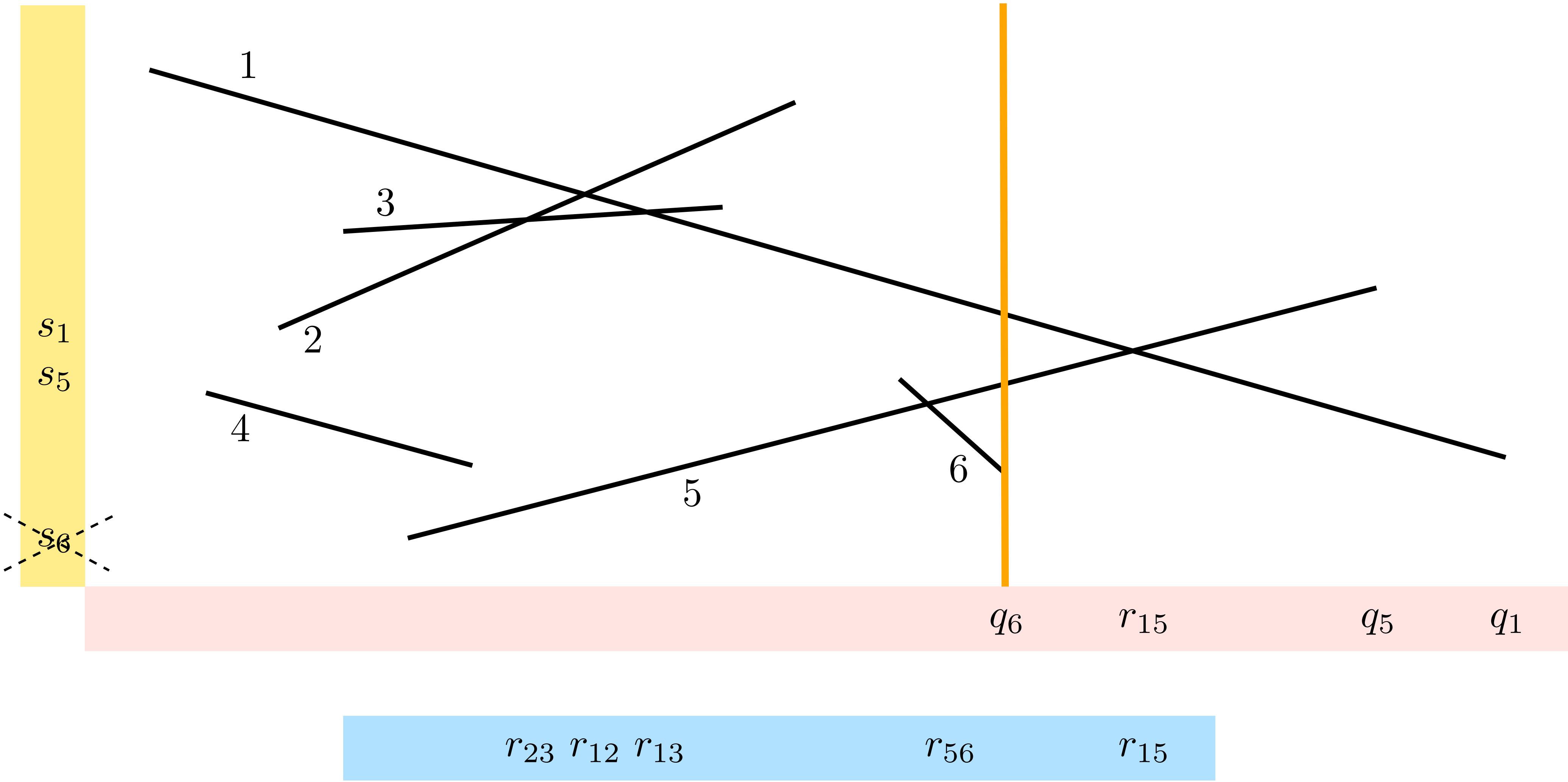
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

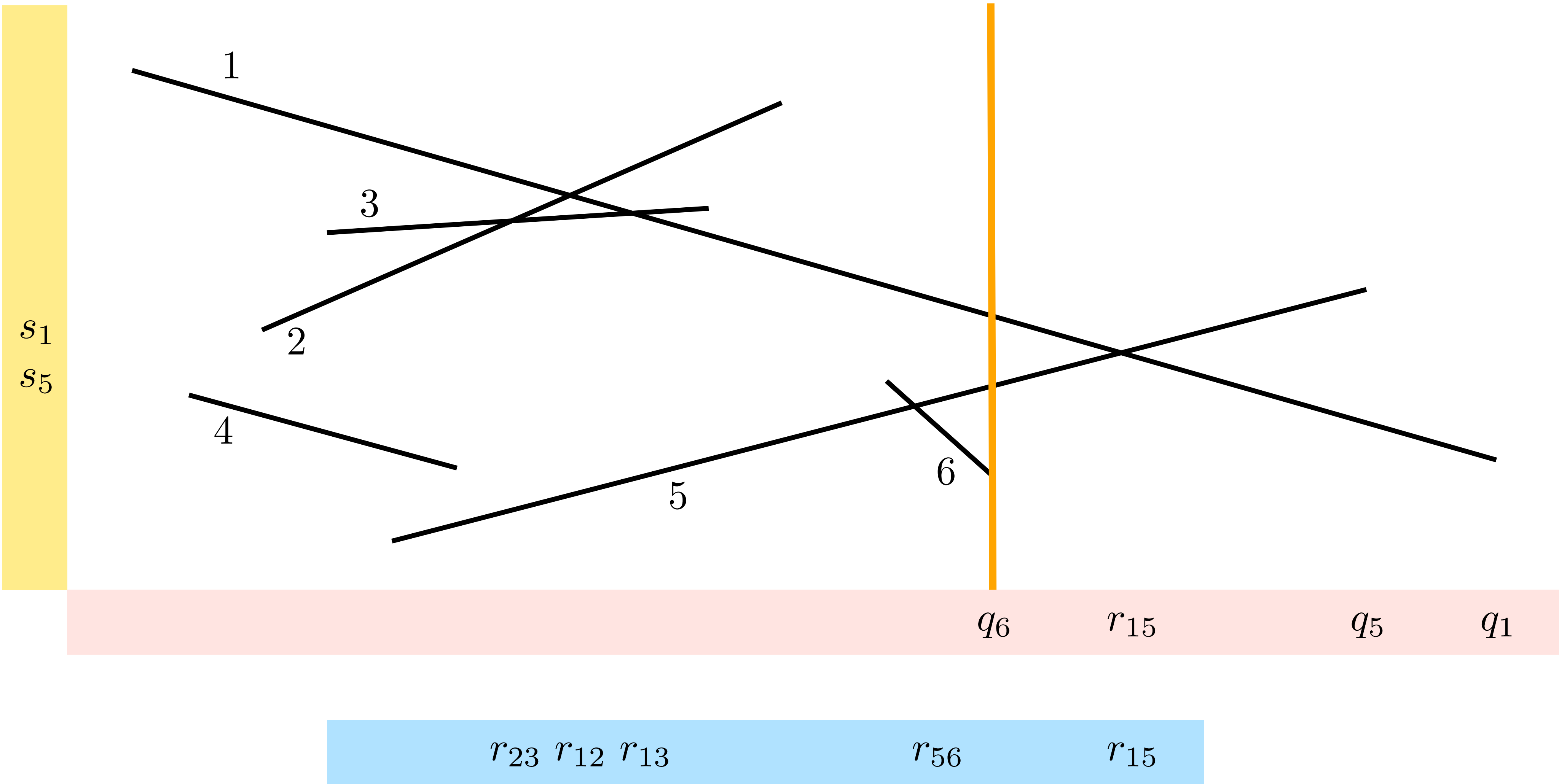


Bentley-Ottman's algorithm: simulation

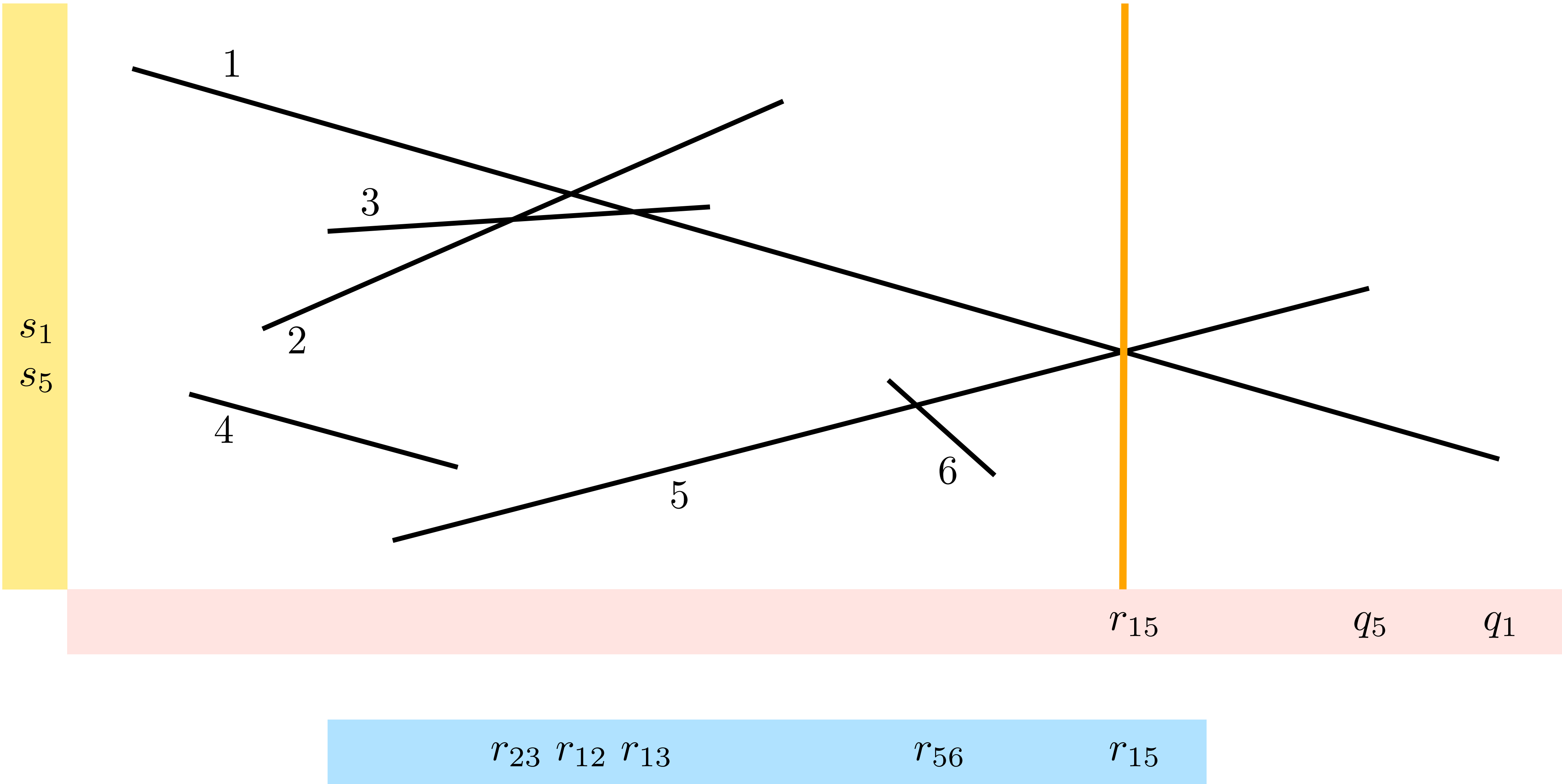


INTERSECTION OF LINE-SEGMENTS

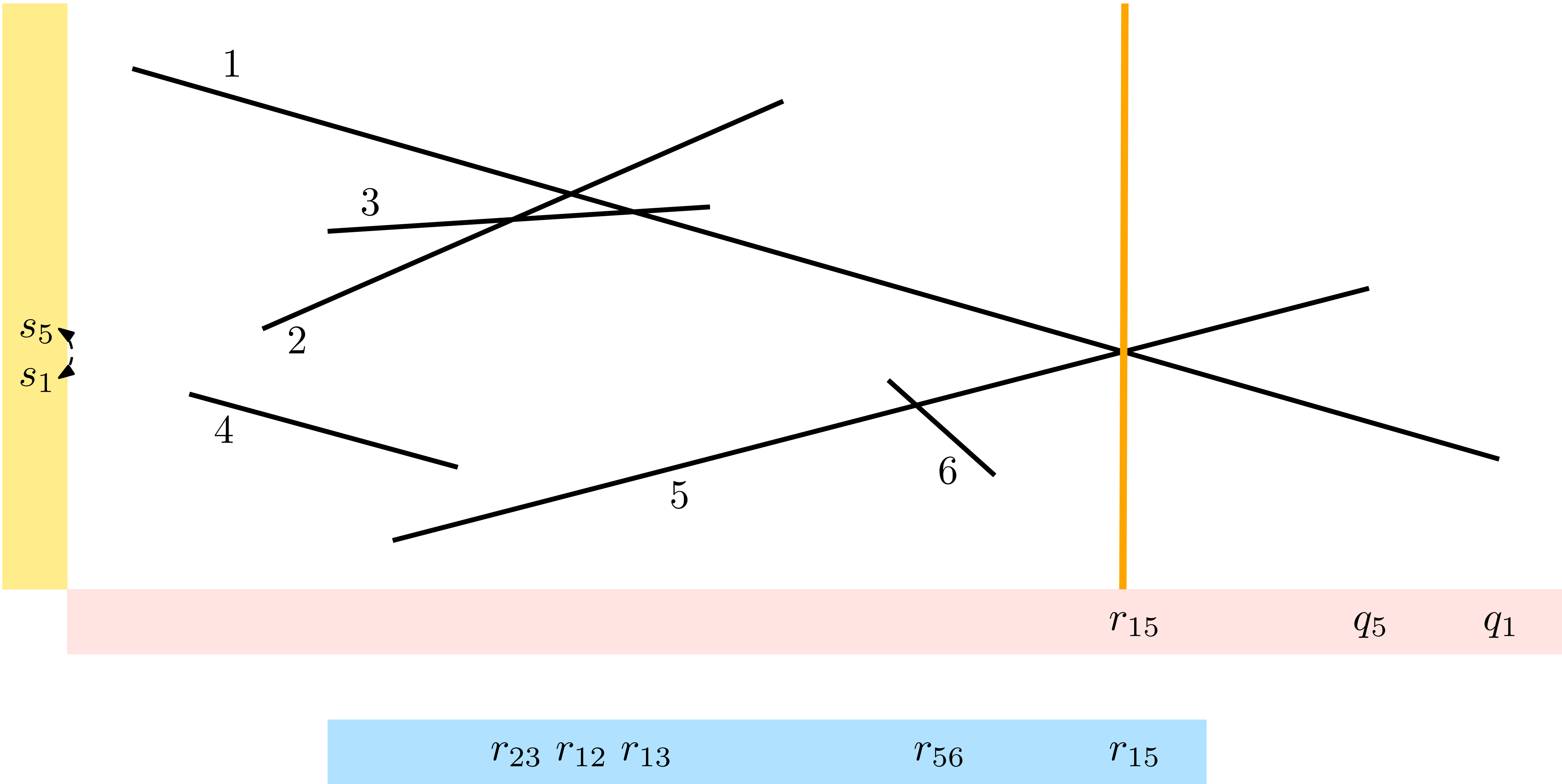
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

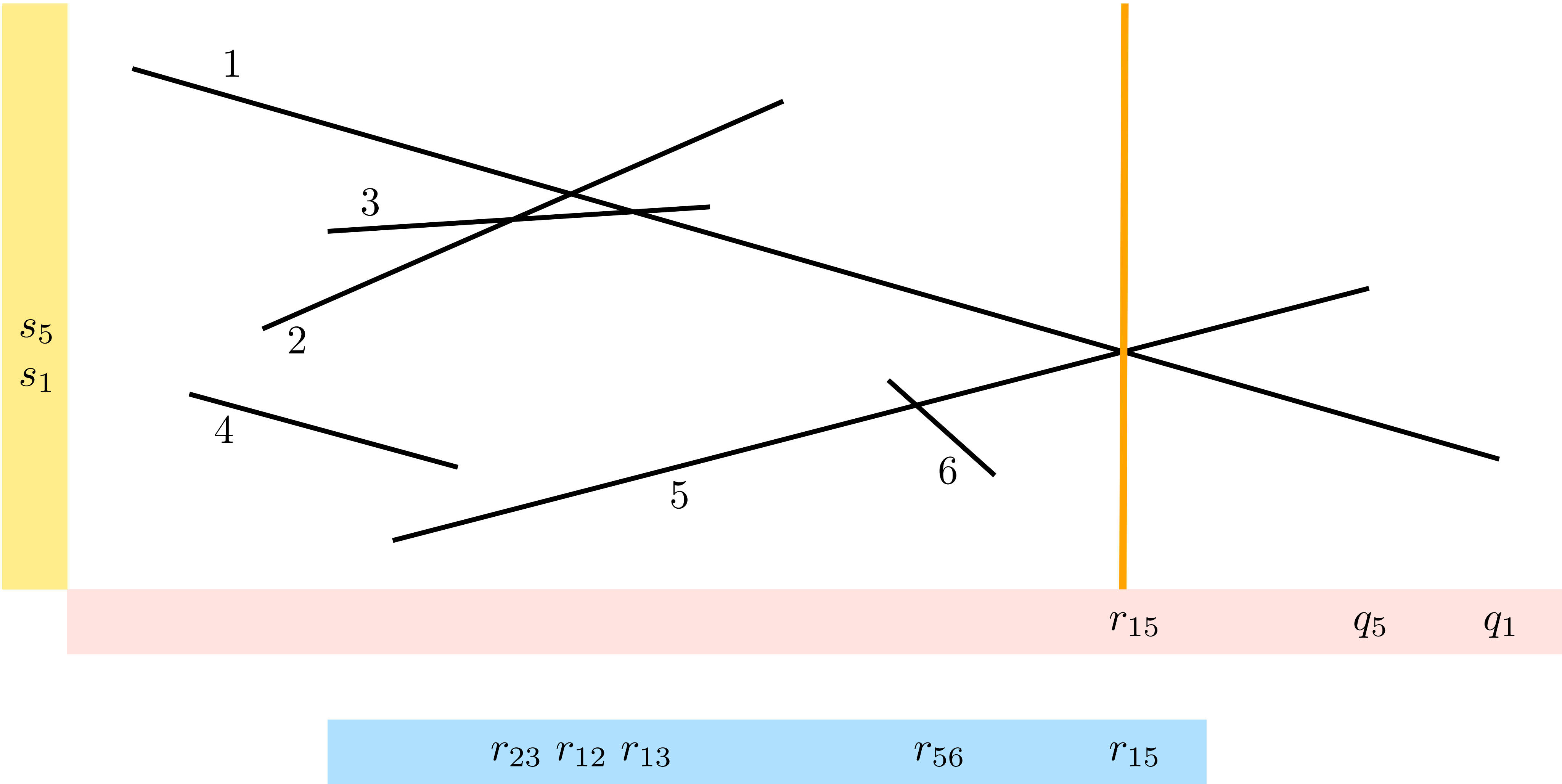


Bentley-Ottman's algorithm: simulation

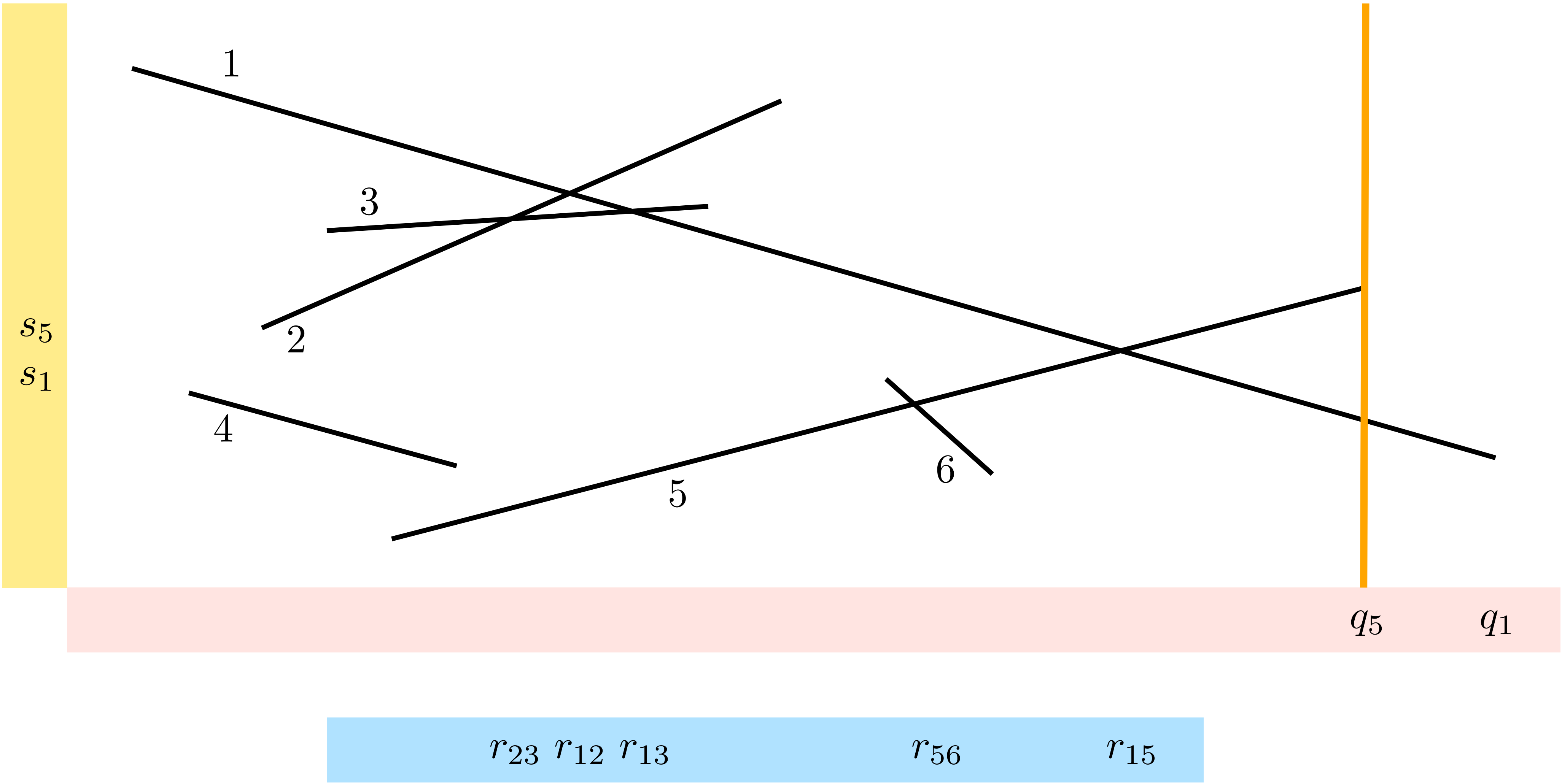


INTERSECTION OF LINE-SEGMENTS

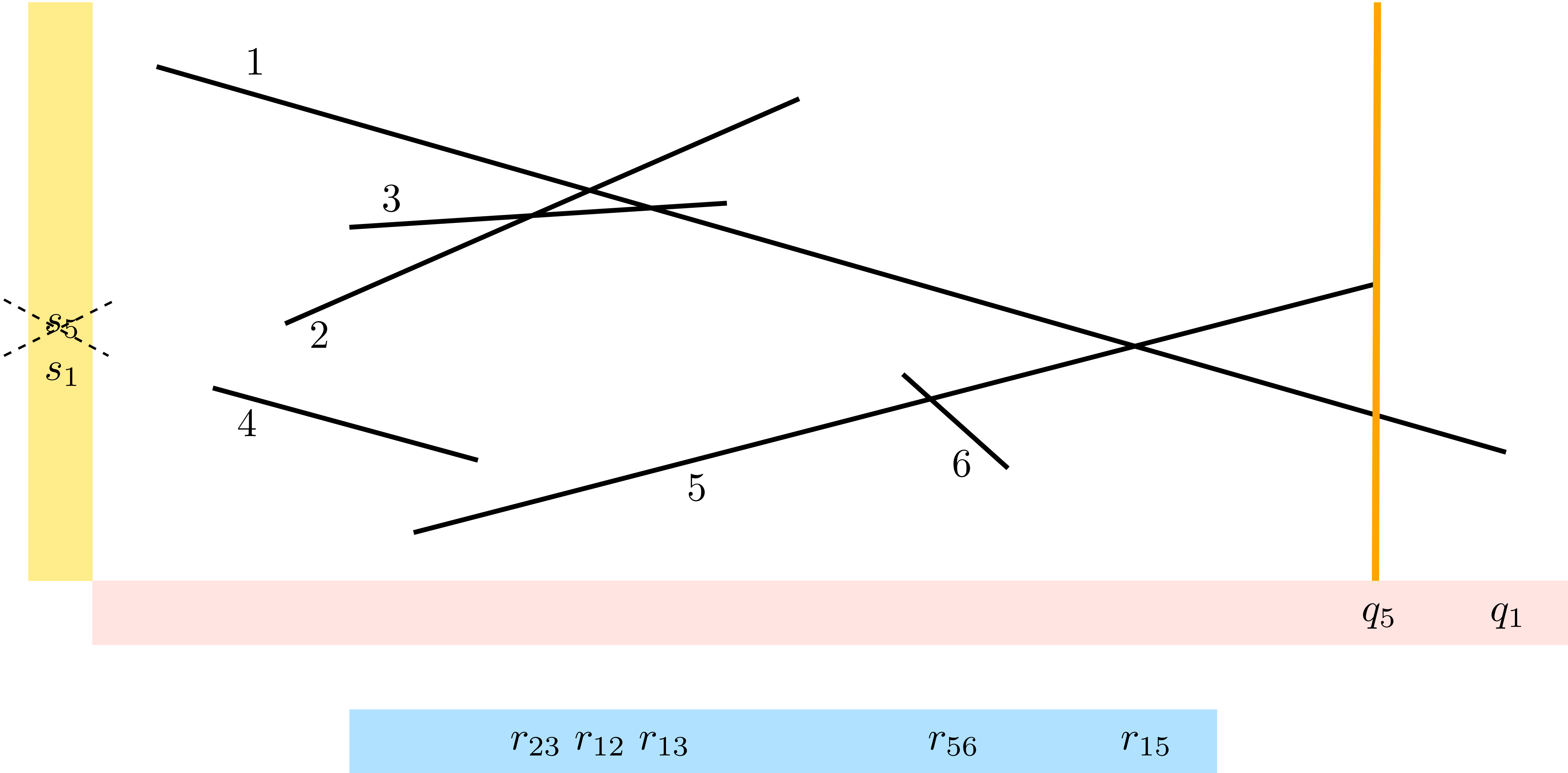
Bentley-Ottman's algorithm: simulation



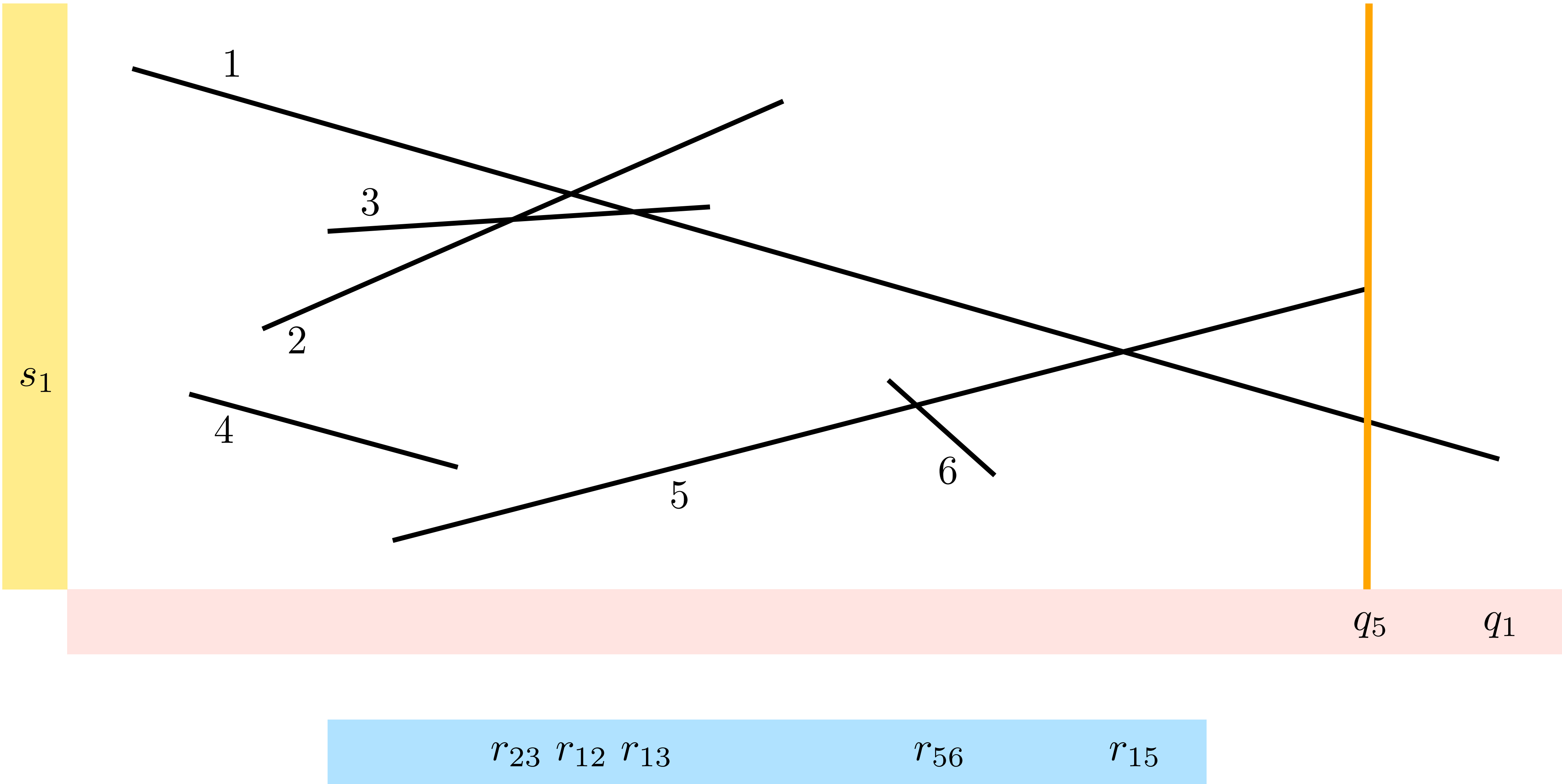
Bentley-Ottman's algorithm: simulation



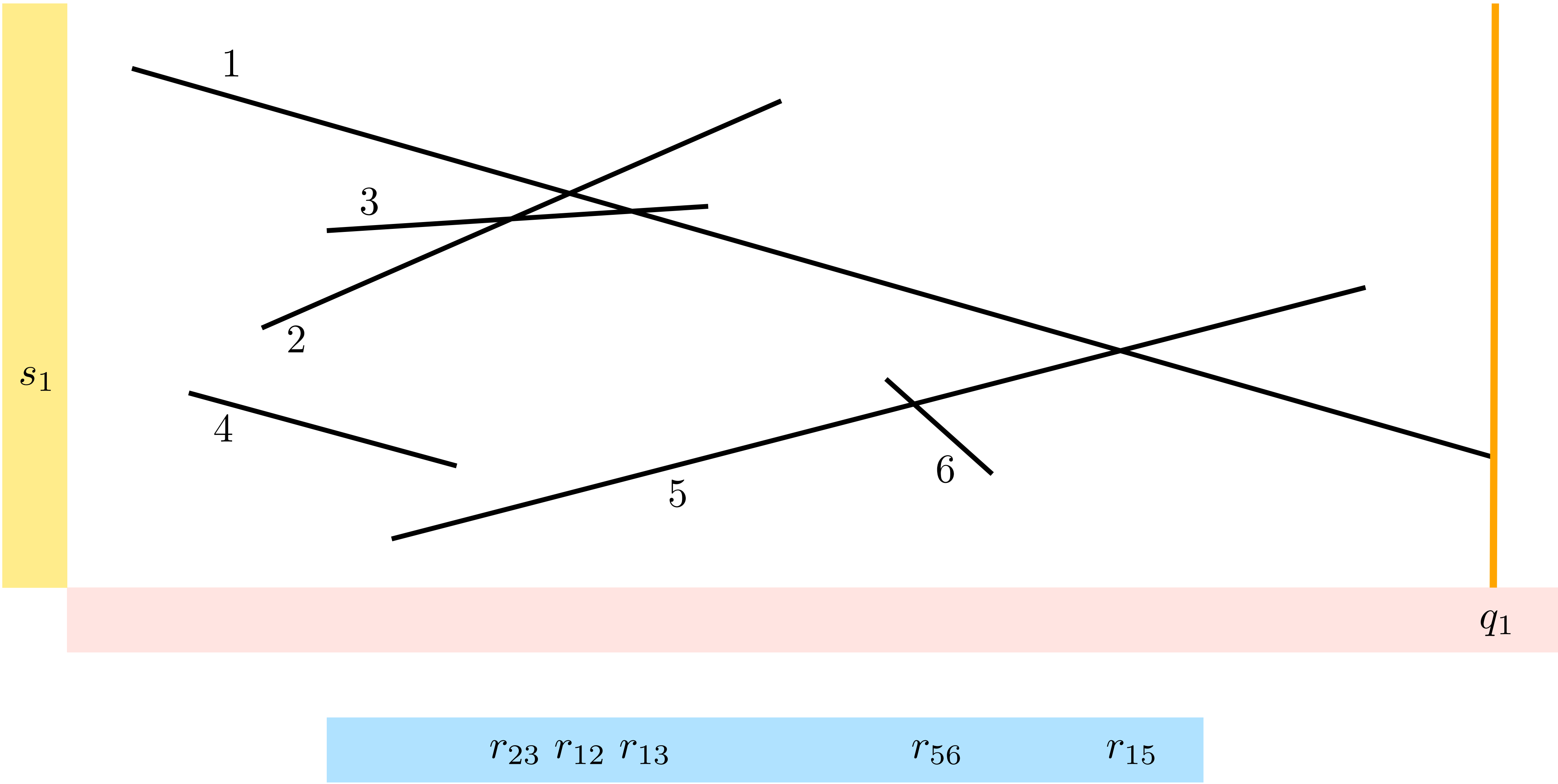
Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation

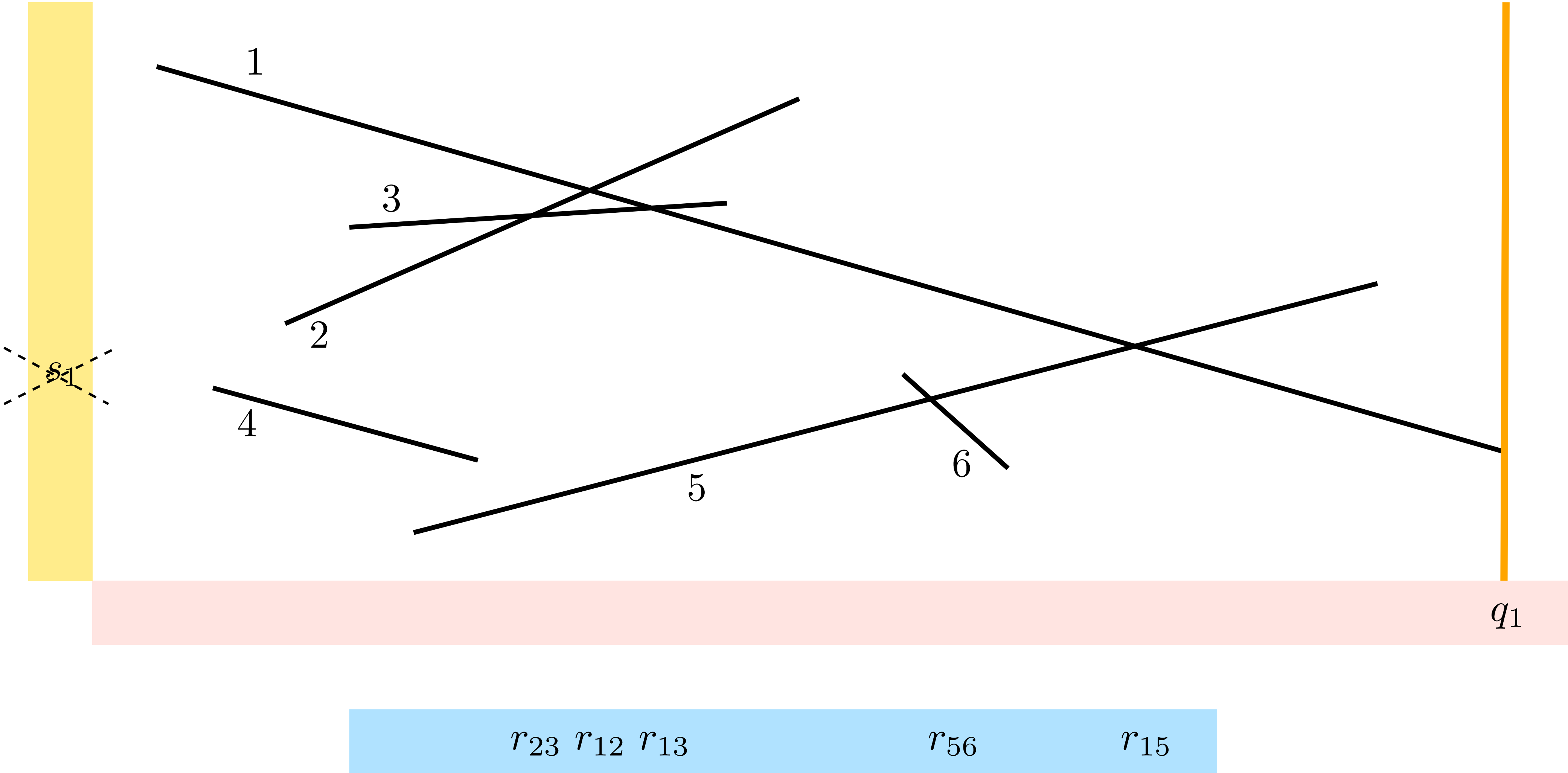


Bentley-Ottman's algorithm: simulation

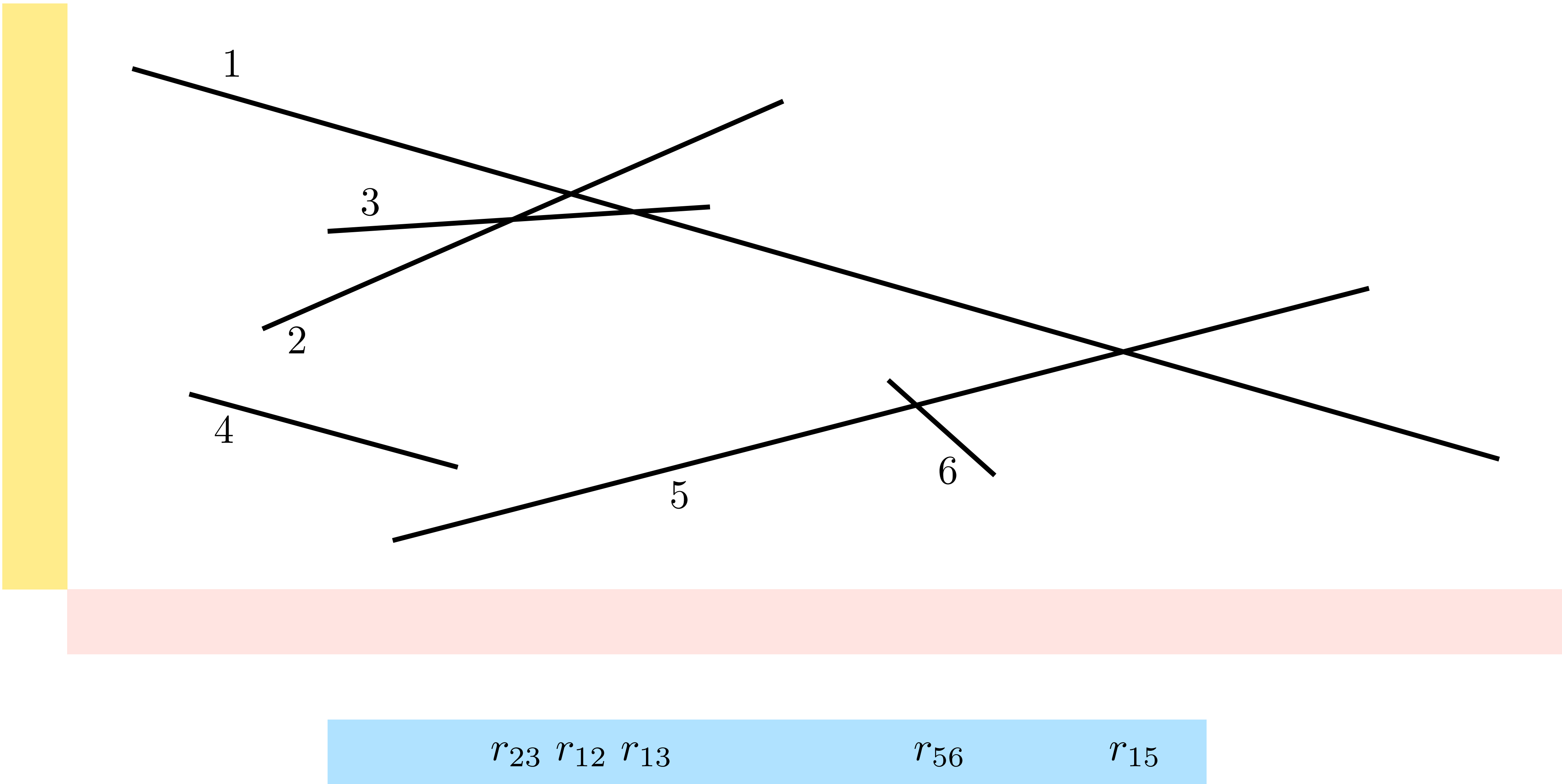


INTERSECTION OF LINE-SEGMENTS

Bentley-Ottman's algorithm: simulation



Bentley-Ottman's algorithm: simulation



Bentley-Ottman's Algorithm

Bentley-Ottman's Algorithm

Correctness

- The algorithm finds all intersections (due to Observation 2).
- The algorithm does not find any faked intersection (all intersections are checked).

Bentley-Ottman's Algorithm

Correctness

- The algorithm finds all intersections (due to Observation 2).
- The algorithm does not find any faked intersection (all intersections are checked).

Dealing with degenerate cases

- In order to deal with input data containing more than one point sharing the same abscissa, the event queue E must store the points in lexicographical order (and not only by abscissae).
- The algorithm can trivially detect whether two or more line-segments intersect in more than one point (i.e., intersect in a line-segment), since it stops at their endpoints.
- A slight modification also allows to deal with input data in which three or more line-segments intersect at the same point: in this case, the algorithm inverts their order in the sweep line at the intersection point event.

Bentley-Ottman's Algorithm

Data structures

Bentley-Ottman's Algorithm

Data structures

Sweep line, L:

Keeps the total order of the stabbed line-segments and supports:

- insert(s)
- delete(s)
- transpose(s_1, s_2)
- previous(s)
- next(s)

A dictionary (balanced binary tree) allows to perform each of these operations in $O(\log n)$ time.

Bentley-Ottman's Algorithm

Data structures

Events queue, E:

Keeps the total order of the events and supports:

- minimum (report and extract)
- insert(p)
- memberQ(p)

A priority queue (balanced binary tree) allows to perform each of these operations in $O(\log n)$ time.

Bentley-Ottman's Algorithm

Complexity (time)

Bentley-Ottman's Algorithm

Complexity (time)

Initialization: $O(n \log n)$

Advance (performed $2n + k$ times):

1. $O(\log n)$
2. 3. 4. $O(\log n)$
5. $O(\log n)$

Total: $O((n+k) \log n)$

Initialization:
- Sort the $2n$ endpoints by abscissa and store the information in E.
- Line L starts empty.

Advance

While $E \neq \emptyset$ do:

1. $p = \min E$
2. If $p = \text{start}(s)$, then...
3. If $p = \text{end}(s)$, then...
4. If $p = s_1 \cap s_2$ with $s_1 <_L s_2$, then...
5. Delete p from E

Bentley-Ottman's Algorithm

Complexity (time)

Initialization: $O(n \log n)$

Advance (performed $2n + k$ times):

1. $O(\log n)$
2. 3. 4. $O(\log n)$
5. $O(\log n)$

Total: $O((n + k) \log n)$

The previous counting corresponds to the non degenerated case.
When each intersection point, v_i, may correspond to more than two intersecting line-segments, the total running time of the advance step of the algorithm is $O((\sum_{i=1}^{k} \mathrm{degree}(v_i)) \log n)$.
However, considering the points v_i as vertices of the graph:

$$\sum_{i=1}^{k} \mathrm{degree}(v_i) \leq 2e = O(e) = O(v) = O(2n + k) = O(n + k).$$

Bentley-Ottman's Algorithm

Complexity (space)

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line-segments.

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line-segments.

In the formulation exposed so far, the event queue may at some point store all intersection points, which are $k = O(n^2)$.

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line-segments.

In the formulation exposed so far, the event queue may at some point store all intersection points, which are $k = O(n^2)$.

However, a slight modification allows the event queue to store, at each step of the algorithm, at most $n-1$ intersection events. This can be achieved if, at each step, E only stores intersection points of line-segments adjacent in L, and the intersection points are deleted from E whenever the intersecting segments become non adjacent.

The decision problem

The decision problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: there is / there is not a pair of intersecting line-segments, and report a witness.

INTERSECTION OF LINE-SEGMENTS

The decision problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: there is / there is not a pair of intersecting line-segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

The decision problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \ldots n$.
Output: there is / there is not a pair of intersecting line-segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

Lower bound

The decision problem has complexity $\Omega(n \log n)$.

INTERSECTION OF LINE-SEGMENTS

The decision problem

Input: n line-segments in the plane, $s_i = (p_i, q_i),\ i = 1 \dots n$.
Output: there is / there is not a pair of intersecting line-segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

Lower bound

The decision problem has complexity $\Omega(n \log n)$.

Proof: by reduction from unicity of integers.

Given $x_1, \dots, x_n \in \mathbb{N}$, compute $p_i = (x_i, 0)$, $q_i = (x_i, 1)$ and $s_i = (p_i, q_i)$.
There exists a pair of intersecting line-segments if and only if there exist duplicate numbers in the original set.

If you don't like degeneracies, consider the following points:
$p_i = (x_i - \frac{1}{2i}, 0)$ and $q_i = (x_i + \frac{1}{2i}, 1)$.

The problem of reporting all intersections

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n \log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n \log n)$ time

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n \log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n \log n)$ time

Optimal algorithm

An algorithm by Chazelle and Edelsbrunner solves this problem in $\Theta(k + n \log n)$ time.

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n\log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n\log n)$ time

Optimal algorithm

An algorithm by Chazelle and Edelsbrunner solves this problem in $\Theta(k + n\log n)$ time.

Consequences

- Deciding whether a polygon is simple can be solved in $O(n\log n)$ time.
- Deciding whether two simple polygons intersect can be solved in $O(n\log n)$ time.

FURTHER READING

- F. Preparata, M. Shamos, *Computational Geometry: An introduction*, Springer.
- M. de Berg, O. Cheong, M. van Kreveld, M. Overmars: *Computational Geometry: Alhorithms and Applications*, Springer.
- J-D. Boissonnat, M. Yvinec, *Algorithmic Geometry*, Cambridge University Press.